A presente dissertação foi realizada no âmbito do curso de Mestrado em Engenharia Informática, no Departamento de Informática da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa[1]. A dissertação foi realizada com o enquadramento e orientação do Prof. Doutor Henrique João Lopes Domingos, tendo sido desenvolvida no Grupo de Investigação de Sistemas Distribuídos de Grande Escala do CITI[2] – Centro de Informática e Tecnologias da Informação.

A temática da dissertação envolve o estudo e análise de modelos e suportes de segurança para sistemas de computação distribuída com ênfase nos sistemas vocacionados para suporte de comunicação fiável orientada para grupos. Estes sistemas são vulgarmente conhecidos por "*SCG - Sistemas de Comunicação em Grupo* ou *SCFG - Sistemas de Comunicação Fiável em Grupo*" (ou GCS - *Group Communication Systems* RGCS ou *RGCS - Reliable Group Communication Systems*, na terminologia em língua inglesa). Ao longo da dissertação a sigla SCG será usada preferencialmente como referência para este tipo de sistemas.

Título da dissertação:

[1] Mais informações sobre o DI-FCT-UNL podem ser encontradas em http://www.di.fct.unl.pt

[2] O CITI – Centro de Informática e Tecnologias da Informação é um Centro de Investigação sedeado no DI-FCT-UNL. Mais informação sobre o Centro e suas actividades pode ser encontrada em http://www.di.fct.unl.pt/citi.

Autenticação, Controlo de Acessos e Confidencialidade em Sistemas de Comunicação Fiável em Grupo

Autor:

Bruno Miguel de Almeida Bernardes

Palavras-chave:

Comunicação multi-ponto

Comunicação em grupo

Comunicação fiável em Grupo

Sistemas para Suporte de Comunicação Fiável em Grupo

Sistemas Distribuídos

Propriedades e Serviços de Segurança em Sistemas Distribuídos

Autenticação

Confidencialidade

Controlo de acessos

Keywords:

Multicast Communication

Group Communication

Reliable Group Communication

Reliable Group-Oriented Communication Support Systems

Distributed Systems

Security Properties and Services in Distributed Systems

Authentication

Confidentiality

Access Control Mechanisms

Agradecimentos

Agradeço à Neoris Portugal por me ter sido proporcionado o tempo e disponibilidade possíveis para o desenvolvimento dos trabalhos relativos à presente dissertação.

Agradeço ao Departamento de Informática da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, por me terem sido proporcionados recursos, meios e acesso às instalações, não só no âmbito dos trabalhos directamente associados ao desenvolvimento específico da presente dissertação, mas igualmente durante a realização da parte escolar do curso de Mestrado.

Agradeço ao Prof. Doutor Henrique João Lopes Domingos, pelas pistas que me indicou no âmbito da sua orientação e pela paciência e disponibilidade face às minhas dificuldades de disponibilidade e atrasos nos prazos.

Agradeço ainda a toda minha família, em especial aos meus pais pelo carinho e suporte demonstrado durante estes últimos meses e principalmente à minha companheira, Ana Lopes, pela sua força de vontade, paciência e comentários para a elaboração desta tese. Um carinho para especial para o João Monteiro e Susana Cabaço pelo seu incentivo.

Sumário

A investigação na área dos sistemas de comunicação fiável em grupo (*Reliable Group Communication Systems* ou RGCS) produziu resultados relevantes durante a segunda parte da década de 1990. Estes sistemas permitem suportar, de forma eficiente e fiável, requisitos de comunicação em grupo, tendo em vista suportar aplicações orientadas para grupos de processos numa perspectiva intra-grupo (explorando modos de comunicação ponto-a-ponto e ponto-multiponto entre processos filiados num grupo) ou inter-grupo (explorando modos de comunicação ponto-multiponto e grupo-a-grupo entre processos filiados em grupos distintos). Complementarmente, os sistemas de comunicação fiável em grupo disponibilizam os necessários mecanismos de coordenação e controlo da filiação e gestão dinâmica da participação de processos em grupo, assegurando o suporte necessário para concepção de sistemas distribuídos baseados em computações multi-participadas.

O suporte de segurança para sistemas de comunicação fiável em grupo foi um aspecto inicialmente negligenciado na investigação e materialização de sistemas de comunicação fiável em grupo. A investigação focalizou-se inicialmente na concepção, optimização e integração dos protocolos de comunicação, nos mecanismos de gestão e controlo dinâmico de participação ou filiação de processos em grupo e nos aspectos associados ao suporte de fiabilidade daqueles protocolos face a modelos de falhas. Só mais recentemente os mecanismos e serviços de segurança emergiram de forma mais marcante na investigação desses sistemas.

A presente dissertação aborda a problemática de adicionar e integrar serviços de segurança em sistemas de computação orientados para grupos. O objectivo é o estudo e a caracterização particular de propriedades e suportes para autenticação em grupo, controlo de acessos e confidencialidade em protocolos de comunicação fiável em grupo. A partir deste estudo propõe-se um modelo de integração de um serviço de autenticação, mecanismo de controlo de acessos e gestão e distribuição de chaves criptográficas, para sessões seguras de comunicação fiável em grupo. A integração destes serviços é perspectivada a partir de uma camada de segurança de comunicação em grupo (SGCL - *Secure Group Communication Layer*). Esta camada cria uma abstracção do sistema para fornecer um serviço fiável e seguro para gestão e suporte de aplicações baseadas em computações orientadas para grupos.

Abstract

The research on Reliable Group Oriented Communication Systems (RGCSs) produced important results in the second part of the last decade. These systems and their protocols are designed to support reliable and efficient intra-group communication (combining unicast or multicast communication models) or inter-group communication (combining multicast or group-to-group communication models). In addition, a RGCS offers coordination mechanisms to support the notion of process-group and membership control services to support and manage multi-process group oriented distributed computations. Different toolkits and programming emerged from that research, helping programmers in the development of different distributed applications with requirements involving: reliable multicasting communication, fault-tolerance, replication, support for cooperative-oriented interactions, group-oriented collaboration and awareness control, etc.

The initial research on those systems, neglected, in general, the support for security properties and services. In the other hand, the usual security services used in distributed systems have been concentrated on setting up secure communication channels involving two parties (pairs of principals) in typical client/server communication models. In group-oriented distributed computing systems, it is often necessary to enable secure group-oriented communication channels. This involves the support for authentication, access-control, confidentiality and data integrity supported in reliable multicasting communication and in a group-oriented perspective. A typical example is that of a secure and large-scale replicated server, for which all communication between the distributed replicas should be protected against identity spoofing and modification, fabrication or even abusive passive interception of messages. Other example is that of a secret conference between a closed-group of users distributed along the Internet.

This dissertation is focused in the problematic of adding and integrating group-oriented security services in a reliable group-communication system. The objective is to analyze and characterize the support for group authentication, access control and confidentiality. We propose a model in which a secure group-communication layer (SGCL) bundles a reliable group communication system, integrating a group authentication service, access-control mechanisms and a key-management and distribution protocol. The design and implementation issues of a prototype based on such model allowed in the discussion of other open and challenging issues that can be addressed as future research directions.

Índice

Lista de figuras

Lista de tabelas

Capitulo 1 - Introdução

Este capítulo apresenta um enquadramento inicial, a motivação, o âmbito e os objectivos da dissertação. Inicialmente é introduzida a problemática da segurança na perspectiva dos sistemas de comunicação em grupo e as vertentes envolvidas. Seguidamente faz-se o enquadramento da motivação e dos objectivos da dissertação, referindo-se o trabalho realizado e as principais contribuições da dissertação. Na parte final é apresentada a organização e o conteúdo dos restantes capítulos da dissertação.

1.1. Enquadramento da dissertação

A investigação na área dos sistemas de comunicação fiável em grupo – (*Reliable Group Communication Systems* ou RGCS) ou, mais simplificadamente SCG – Sistemas de Comunicação em Grupo (terminologia adoptada ao longo da dissertação), produziu resultados relevantes que se tornaram muito divulgados na segunda parte da década de 1990. Estes sistemas disponibilizam, de forma eficiente e fiável, suportes adequados face aos requisitos de comunicação envolvendo grupos de entidades participantes. Numa abordagem fundamental de Sistemas Distribuídos, este suporte corresponde primariamente à visão de comunicação fiável em modo multi-ponto, envolvendo grupos de processos. O suporte dos sistemas para comunicação em grupo (ou SCG) visam suportar aplicações orientadas para grupos de processos tanto numa perspectiva intra-grupo (explorando modos de comunicação ponto-a-ponto e ponto-multiponto entre processos filiados num grupo) ou inter-grupo (explorando modos de comunicação ponto-multiponto e grupo-a-grupo entre processos isolados e processos filiados em grupos distintos). Complementarmente, os sistemas de comunicação fiável em grupo disponibilizam mecanismos de coordenação e controlo da filiação e de gestão dinâmica da participação dos processos em grupos, assegurando o suporte fundamental para concepção, implementação e operação de sistemas distribuídos baseados em computações multi-processos ou multi-participadas.

O suporte de segurança para sistemas de comunicação fiável em grupo foi um aspecto inicialmente negligenciado na investigação e materialização dos sistemas de comunicação fiável em grupo. A investigação inicial e os interesses da comunidade de investigação neste tipo de sistemas, focalizaram-se sobretudo na concepção, optimização e integração de protocolos para comunicação fiável em grupo[3], nos

[3] Na terminologia e bibliografia em língua inglesa, estes protocolos são vulgarmente designados por "*reliable multicasting protocols*" ou "*reliable group-communication protocolos*".

mecanismos de gestão e controlo dinâmico de participação ou filiação de processos em grupo e nos aspectos associados a diferentes dimensões do suporte de fiabilidade daqueles protocolos face a modelos de falhas. Complementarmente, a abordagem da estruturação de ambientes de programação baseados nos suportes anteriores foi igualmente uma das direcções da investigação nos sistemas para comunicação fiável em grupo. Só mais recentemente os mecanismos e serviços de segurança emergiram de forma mais marcante na investigação desses sistemas.

A investigação na área dos SCG produziu assim, durante a década de 1990, resultados relevantes, destacando-se alguns sistemas de referência, como por exemplo: ISIS [48] TOTEM [49], HORUS [41], TRANSIS [50], ENSEMBLE [57], SPREAD [27], INTERGROUP [51], RELACS [52], TIMEWHEEL [53], PHOENIX [54].

Muitos dos anteriores sistemas disponibilizam *toolkits* e ambientes integrados de programação, bem como suportes para execução, possibilitando aos programadores o desenvolvimento simplificado de aplicações que possam beneficiar de modelos de computação orientados para grupos, sem que tenham que conhecer e "confrontar-se" com os detalhes associados à problemática e à complexidade dos suportes de comunicação em grupo ao nível de serviços de *middleware* ou de suportes de sistema distribuído. No entanto, a perspectiva de desenvolvimento de suportes para propriedades de segurança encontra-se descurada em grande parte dos referidos sistemas.

Na presente dissertação perspectiva-se o estudo da problemática de adicionar e integrar serviços de segurança no âmbito dos suportes dos sistemas de computação orientados para grupos. Partindo do estudo e caracterização particular de propriedades e suportes para autenticação em grupo, controlo de acessos em grupo e confidencialidade em contextos de comunicação em grupo, pretende-se perspectivar a integração de serviços de segurança que forneçam suporte a essas propriedades. A abordagem privilegia uma visão de integração de serviços de autenticação, mecanismo de controlo de acessos e suporte de gestão e distribuição de chaves criptográficas, tendo em vista fornecer um suporte para estabelecimento e manutenção de sessões seguras de comunicação fiável em grupo. A integração destes serviços tem em vista a concretização de uma camada de segurança de comunicação em grupo (SGCL - *Secure Group Communication Layer*). Os serviços desta camada devem permitir criar uma abstracção para desenvolvimento e suporte de aplicações seguras baseadas em computações orientadas para grupos.

1.2. Relevância dos Sistemas de Comunicação em Grupo

Verifica-se hoje um interesse crescente e um incremento no número e na diversidade de aplicações e serviços distribuídos que utilizam modos de comunicação em grupo (ou, mais genericamente, modos de comunicação multi-ponto). Aplicações para vídeo-conferência em grupo, aplicações com requisitos de disseminação de informação por grupos, sistemas de suporte de emissão de vídeo a pedido (ou *video-on-demand*), aplicações de *groupware* ou de suporte de trabalho cooperativo (ou *CSCW – Computer Supported Cooperative Work* na terminologia em língua inglesa), aplicações para suporte de decisão em grupo, jogos

em ambientes multi-utilizador, *software* para ambientes de colaboração científica, simples ferramentas de colaboração do tipo "quadros brancos", editores cooperativos, sistemas de votação, sistemas de mensagens em grupo, etc., ou ainda aplicações no domínio de bases de dados replicadas e tolerantes a falhas, são exemplos de aplicações que se adequam naturalmente a modos de comunicação orientados para grupos.

Numa parte significativa dos casos anteriores, a utilização do tipo de suporte convencional dos sistemas de comunicação em grupo permite que as aplicações possam beneficiar de propriedades de fiabilidade para suporte de interacções de características síncronas (ou de tempo real) entre grupos de processos ou entre entidades principais envolvidas[4]. Por outro lado, muitas dessas aplicações requerem mecanismos de coordenação associados à gestão da participação dos principais em grupo, sendo essa gestão assente na visão ou vista consistente das outras entidades que participam no grupo em cada instante. Este tipo de requisitos são complexos de endereçar de forma *ad-hoc* ou caso a caso, principalmente quando se perspectivam conjuntamente com propriedades de escala e de tolerância a falhas. Note-se que o ponto de vista dos critérios de escala envolve só por si dimensões variadas, desde o ponto de vista das características do suporte de comunicação de base, abrangência da distribuição das entidades principais e recursos partilhados ou no número e granularidade dessas entidades e recursos. Os sistemas de comunicação em grupo fornecem o suporte adequado para uma visão suportada em serviços genéricos que atendem a esse tipo de requisitos, residindo esta perspectiva no estabelecimento da noção de "grupo de processos" (ou *process-groups*[5]).

Para o estabelecimento da noção de grupo de processos, um sistema de comunicação em grupo (SCG) disponibiliza um conjunto de protocolos de comunicação multi-ponto (normalmente organizados numa pilha). Estes protocolos permitem complementar o suporte básico de comunicação ponto-a-ponto ou multi-ponto não fiável, ao nível do suporte básico de comunicação de um sistema distribuído convencional, com modos de comunicação fiável e multi-ponto para grupos de processos. Os protocolos disponibilizados num SGC podem materializar diferentes semânticas de comunicação fiável em grupo. Estas semânticas correspondem a diferentes critérios e garantias de fiabilidade subjacentes ao envio e entrega de mensagens bem como a diferentes critérios de ordenação das mensagens. Adicionalmente os SCG disponibilizam mecanismos integrados de organização e manutenção da filiação de processos em grupos, disponibilizando

---

[4] Como é habitual no estudo da segurança em *Sistemas Distribuídos* e *Redes de Computadores*, o termo "principal" é utilizado para referir uma entidade designada do sistema. A noção de principal, dependendo do nível de abordagem ou do nível de estruturação dos suportes e serviços do Sistema Distribuído, pode corresponder a diferentes entidades, a saber:; dispositivos de comunicações ou computadores designados por endereços do nível ligação de dados ou rede, *endpoints* de comunicação de processos (ao nível do transporte de mensagens), componentes ou objectos de aplicações (com designação global no sistema) ou mesmo utilizadores (designados por algujm tipo de identificação digital). No âmbito da dissertação e no contexto de utilização específica do termo "principal", o nível de abordagem dessas entidades será clarificado.

[5] O termo "process-group" é utilizado correntemente, na área dos sistemas de comunicação fiável em grupo, para designar uma entidade principal associada a um grupo de processos. Neste contexto o sistema deve ser capaz de garantir o necessário suporte de designação desse tipo de entidades, o suporte de comunicação em grupo, eventualmente com diferentes semânticas possíveis para comunicação fiável em grupo e os mecanismos de gestão e controlo da organização dos grupos, nomeadamente o controlo da participação ou filiação de processos em grupos de processos.

um suporte coerente para estabelecimento da noção de *vista dos grupos*, representando esta noção o suporte para uma visão coerente e sincronizada com o suporte de comunicação, da participação dinâmica dos processos em grupos. A gestão das vistas dos grupos de processos é dinâmica, suportando entradas e saídas de processos nos grupos durante o ciclo de vida dos grupos e do contexto de comunicação em grupo. Os mecanismos de gestão da participação dos processos nos grupos permitem que se estabeleça um princípio de coordenação e organização dos grupos, como um serviço de gestão dinâmica da filiação dos processos nos grupos[6].

Este serviço permite que os processos possuam, em cada instante, uma visão (ou vista) coerente da constituição dos grupos. Uma vez que do ponto de vista do suporte de execução computacional os processos representam contextos de execução de entidades principais do sistema distribuído, as aplicações suportadas num SCG podem beneficiar desse serviço, observando de forma consistente os eventos associados a entradas ou saídas de processos no grupo. Desta forma, é possível assegurar-se uma gestão coerente de um contexto de multi-participação, independentemente do nível de abstracção associado aos principais do sistema distribuído (desde o nível sistema, onde as entidades principais correspondem a processos em execução e seus pontos finais de comunicação (ou *communication endpoints*) até ao nível das aplicações, onde as entidades principais podem estar associadas a componentes ou recursos das aplicações ou a representações de utilizadores participantes em grupos de utilizadores.

Dadas as características de tolerância a falhas dos SGC, os eventos de entrada e saída de principais e a notificação da sua ocorrência são geridos independentemente das condições operacionais normais ou da ocorrência de falhas eventuais ao nível dos processadores, ao nível do suporte de execução desses processos ou ao nível de falhas do meio de comunicação.

Assim, sumariamente, as características associadas aos SCG fazem com que estes sistemas sejam particularmente adequados à concepção e implementação de diferentes tipos de aplicações, com exigências de suporte de propriedades tais como: tolerância a falhas, gestão dinâmica de recursos replicados, distribuição de carga, requisitos de disseminação fiável de mensagens, assegurando ao mesmo tempo critérios de controlo de coerência do agrupamento de entidades e recursos no momento do envio e recepção de mensagens.

1.3. A abordagem de suportes de segurança para SCG

Durante a segunda metade da década de 1990, a investigação associada aos SCG privilegiou sobretudo as seguintes direcções:

- Integração e sistematização dos suportes anteriormente introduzidos em *toolkits* e ambientes de programação capazes de permitirem aos programadores o desenvolvimento de aplicações

[6] A noção de "membership management" associada ao suporte de base fornecido pelos sistemas de comunicação fiável em grupo traduz, na terminologia e bibliografia desses sistemas a noção aqui introduzida.

complexas do ponto de vista dos requisitos referidos, sem que estes tivessem que conhecer a complexidade e detalhe dos suportes associados;

- Concepção e desenvolvimento de protocolos de comunicação fiável em grupo, tendo em vista diferentes características e critérios de fiabilidade face a diferentes modelos de falhas;
- Incorporação nos protocolos e serviços dos SGC de diferentes semânticas de fiabilidade e diversos critérios de ordenação de envio e entrega de mensagens;
- Desenvolvimento de novos protocolos e mecanismos de gestão coerente de filiação e controlo de vistas de grupos de processos, através de um estudo das suas reais condições de escalabilidade e possibilidades de optimização e relaxamento controlado dos critérios de consistência da gestão das vistas dos grupos.

Uma grande parte da investigação a partir da segunda metade da década de 1990 foi assim dirigida a aspectos de optimização e desempenho dos protocolos e à problemática de conciliar critérios de desempenho com requisitos associados às garantias subjacentes aos critérios de fiabilidade. Essa investigação, associada ao estudo de condições e critérios de escalabilidade efectiva dos SCG tendo em vista o suporte para mascaramento e recuperação de falhas em cenários de distribuição de processos em redes de grande escala, deu origem a um caderno de encargos complexo. Dada a complexidade envolvida, esse destaque foi inteiramente justificável, tendo em vida cenários de distribuição de processos nos quais o critério de escala se tornou cada vez mais determinante.

Rapidamente os estudos associados às características de escalabilidade dos SCG envolvendo um elevado número de processos distribuídos em infra-estruturas e ambientes de computação de grande escala (como por exemplo, ambientes do tipo Internet) ocuparam um papel de destaque na investigação. Na maior parte dos casos, os aspectos de suporte de segurança foram sendo negligenciados ou até, em muitos casos, completamente ignorados.

Só mais recentemente (nos últimos anos da década de 1990 e primeiros anos do século XXI), a investigação na vertente de segurança dos sistemas de computação em grupo assumiu um papel mais importante, tornando-se mais visível e entrando para o centro da agenda da investigação, tornando-se mesmo dos aspectos mais importantes na continuidade de alguns dos projectos de investigação mais conhecidos ou, no lançamento de novos projectos.

Actualmente, o aumento da quantidade e a maior diversidade de aplicações que se adequam de forma natural ao suporte proporcionado pelos SCG, tem sido acompanhado de um interesse redobrado em relação à necessidade de serviços de segurança que estes sistemas devem disponibilizar. Os requisitos de segurança das aplicações e a importância crescente que os suportes de segurança adquiriram no âmbito da computação distribuída, justificam esse interesse crescente também na área da investigação dos SCG.

A perspectiva de integração de serviços de segurança para suporte das propriedades de autenticação, confidencialidade, não-repudiação, integridade, controlo de acessos e garantia de elevada disponibilidade no âmbito dos SCG, tendo em vista as particularidades destes sistemas e os necessários serviços de suporte face a tipologias de ataques e à conceptualização e requisitos subjacentes àquelas propriedades

fundamentais de segurança de sistemas distribuídos [60], [61] constitui um desafio na área dos SGC. O suporte dos necessários serviços de segurança, cuja descrição e enquadramento inicial numa visão recente das Redes de Computadores e dos Sistemas Distribuídos podem ser encontrados em diversos autores [65], [Tanembaum 03], torna-se particularmente complexo quando se perspectivam os SCG. Esta complexidade é tanto maior quanto maior é a perspectiva de escala, tendo em vista ambientes computacionais suportados em redes corporativas ou institucionais muito abrangentes e na rede Internet.

1.4. Aproximação aos suportes de segurança para SCG

Embora muitas aproximações iniciais associadas a suportes de segurança em alguns SCG fossem baseadas em técnicas mais ou menos simplistas (como é, por exemplo, o caso do suporte de comunicação privada em grupo por adopção de uma chave criptográfica partilhada pré-configurada estaticamente de forma externa ao contexto de gestão de um grupo de processos), exige-se uma perspectiva integrada das diferentes propriedades de segurança com a problemática da gestão de participação e organização dos grupos e dos protocolos fiáveis para comunicação em grupo. Tal exigência decorre da necessidade de suportar interacções seguras entre grupos de processos em contextos de gestão e participação dinâmica, podendo os processos entrar e sair dos grupos, quer em circunstâncias normais de operação, quer devido às falhas de execução dos processos, dos processadores e do meio de comunicação. Esta aproximação pode revelar-se muito complexa, exigindo desde logo o estudo da adequação aos SCG de métodos e algoritmos da criptografia computacional (tal como são normalmente utilizados na abordagem clássica dos suportes envolvidos nos serviços de segurança dos sistemas distribuídos ponto-a-ponto ou cliente-servidor) [55].

Para efeitos da presente dissertação interessa referir entre essas particularidades os suportes para autenticação em grupo, controlo de acessos em grupo e confidencialidade em grupo.

- No âmbito da autenticação em grupo é necessário garantir que os membros legítimos que participam num grupo se devem identificar e autenticar mutuamente, no âmbito do estabelecimento da noção de vista e de organização do grupo (*group membership / group view*).

- Do ponto de vista de controlo de acessos, é necessário ter em conta que as operações que sejam desencadeadas por membros legítimos do grupo pressupõem um contexto anterior de controlo de acesso no momento da adesão ou entrada de um membro no próprio grupo. Com efeito, a entrada de um novo membro no grupo deve ser decidida com base numa política de controlo de acessos que pode basear-se num processo de autorização em grupo e não necessariamente de uma entidade única.

- Quanto ao suporte para confidencialidade da comunicação em grupo, interessa garantir que a comunicação entre os membros de um grupo permaneça secreta em relação a principais que não façam parte desse mesmo grupo. O contexto de confidencialidade exige métodos de gestão e distribuição de chaves que podem ter requisitos muito diversos do ponto de vista de

propriedades particulares sobre preservação ou não do contexto de confidencialidade num tempo passado à admissão de novos membros ou em momentos futuros à saída de membros. Este aspecto leva-nos a pensar em mecanismos eficientes de gestão, negociação e distribuição dinâmica de novas chaves, de forma sincronizada com o processo de admissão ou de saída de membros de um grupo.

Os serviços de segurança associados à materialização das propriedades de segurança em sistemas distribuídos e suas aplicações (tal como são referenciadas em [60] ou em [61]) e tal como têm vindo a ser estudados, sintetizados e discutidos por diversos autores (ex., [62], [63], [64] ou [65], são normalmente dirigidas a proteger interacções entre pares de entidades principais (sejam interacções entre processos, componentes aplicacionais, processadores ou utilizadores, numa perspectiva de estabelecimento de canais seguros de comunicação ponto-a-ponto e extremo-a-extremo). Tal é o âmbito usual dos serviços de segurança associados às aplicações ou serviços normalmente baseados no modelo cliente/servidor.

Utilizam-se hoje, amplamente, diversos protocolos de base de segurança para âmbitos de computação e suporte de aplicações cliente/servidor. Alguns desses protocolos encontram-se hoje banalizados ao nível das implementações da pilha protocolar TCP/IP (como por exemplo o SSL (ou TLS) [18] ou IPSec [56]). Este tipo de suportes de segurança foram perspectivados para comunicação ponto-a-ponto (processo a processo ou *host* a *host*, respectivamente). Não é imediato que o mesmo tipo de suportes sejam facilmente extensíveis para contextos de comunicação envolvendo grupos de processos (ou modos de comunicação multi-ponto ou ponto-multiponto). Um dos aspectos mais delicados reside no modelo de autenticação (de índole unilateral ou bilateral vocacionado para sessões de interacção entre pares de principais) e no modelo de distribuição de chaves que decorre, quando muito, de um processo de autenticação mútua limitada a dois principais. Por exemplo, em SSL, cada vez que se estabelece uma conexão TCP entre dois processos principais, o processamento consensual de autenticação e o decorrente estabelecimento de chaves criptográficas de sessão (associadas ao estabelecimento da noção de canal de comunicação seguro) inicia-se sem que haja noção de estados anteriores à sessão SSL actual. Se um dos principais termina ou se a conexão falha por qualquer motivo e verificando-se as condições para término da sessão corrente, o outro principal assume a destruição do estado de segurança associado ao canal ponto-a-ponto.

A necessidade de se terem que suportar entradas e saídas dinâmicas de processos num grupo torna a situação anterior completamente diversa. Num SGG é necessário estabelecer a noção de canal seguro para um âmbito de canal de comunicação em grupo. O estabelecimento destes canais envolve o suporte fiável de comunicação com base em garantias de autenticação de todos os membros actuais de um grupo num contexto de sessão segura de grupo (que designaremos por *autenticação em grupo*), confidencialidade da informação trocada no âmbito das interacções desses membros (que designaremos por *confidencialidade em grupo*) e preservação da integridade da informação trocada nessas interacções. É ainda necessário assegurar mecanismos de controlo de acesso perspectivados para modelos de autorizações de admissão de membros que pretendam entrar nos grupos (o que designaremos por *controlo de acesso em grupo*). Desta forma, o contexto de segurança subjacente a uma sessão segura de comunicação em grupo pode subsistir para lá da saída e entrada de novos membros no grupo.

Adicionalmente para se estabelecer um canal seguro de comunicação em grupo, é necessário garantir que eventuais segredos (sejam sementes para cálculo de chaves criptográficas, sejam as próprias chaves criptográficas ou senhas associadas à capacidade de actuar segundo um modelo de autorizações do ponto de vista de controlo de acessos em grupo), sejam partilhados por todos os membros do grupo. Um dos aspectos essenciais para esse suporte é possibilitar que o estabelecimento desses segredos partilhados em grupo (que designaremos genericamente por *gestão e distribuição de chaves de grupo*) decorra de um processo de partilha (que designaremos por *partilha de chaves de grupo*) subjacente às garantias do suporte de autenticação em grupo.

A um mecanismo de distribuição assente num processo de acordo mútuo para geração e cálculo das chaves de grupo, de forma independente por parte de cada membro, designaremos por *acordo de distribuição de chaves em grupo*. Este tipo de mecanismos é particularmente interessante para se perspectivar o suporte de *segurança futura de grupo*, suporte que consiste em que a segurança futura do grupo não dependa de uma entidade distribuidora de chaves que seja externa ao grupo. De igual modo, pode ser exigível que o contexto de segurança passada no âmbito de um grupo deve ser preservada face à admissão e entrada de novos membros para o grupo. Este requisito é designado por *segurança passada de grupo*. Das noções de segurança futura de grupo e segurança passada de grupo decorre a necessidade de se ter que gerir e proceder à distribuição e acordo das chaves em grupo, cada vez que se verifica uma mudança na constituição ou organização do grupo. O mecanismo de partilha de segredos e chaves criptográficas em grupo deve assim ser perspectivado à luz das próprias propriedades de tolerância a falhas dos SCG.

Por outro lado, pode exigir-se que no processo da geração e distribuição de segredos ou chaves em grupo, intervenham obrigatoriamente todos os actuais membros lícitos participantes do grupo. Designaremos este tipo de gestão e distribuição de segredos ou chaves partilhadas num grupo como *gestão* contributiva *das chaves ou segredos do grupo*, enfatizando o carácter de comparticipação e contribuição colectiva de todos os membros de um grupo para o contexto de segurança do grupo[7]. No caso de não se pretender garantir esse tipo de propriedade, a distribuição de chaves aos membros do grupo poderia ser perspectivada a partir de uma entidade centralizada, actuando como centro de distribuição de chaves e externa (e eventualmente autoritária) em relação ao grupo – o que seria um modelo mais próximo do modelo convencional de geração e distribuição de chaves entre entidades principais de um sistema distribuído.

Em qualquer caso, apenas os membros actuais e legítimos de um grupo (de acordo com uma visão síncrona, partilhada e coerente da constituição dos grupos) deverão ter acesso aos segredos partilhados pelo grupo. Refere-se "membros actuais" uma vez que num SGC a gestão da filiação de um grupo é dinâmica, podendo os membros entrar ou sair do grupo em qualquer instante, após a criação do mesmo, o que implica que o contexto de segurança do grupo pode, ele próprio, ser dinâmico. Deste ponto de vista, antecipa-se que a gestão, distribuição e partilha de chaves criptográficas em grupo, pode ter que ser sincronizada e integrada com o mecanismo de gestão da filiação de membros no grupo.

---

[7] Este contexto de segurança é genericamente referido por muitos autores como "*group-secrecy*", na terminologia em língua inglesa das propriedades de segurança em cenários de computação em grupo.

1.5. Propriedades de segurança num SCG

As propriedades de segurança num SCG podem pois ser bastante mais complexas do que num sistema de comunicação envolvendo apenas duas entidades.

O processamento de consenso da vista da filiação do grupo e da autenticação dos membros do grupo exige a comprovação de identidade por parte de todos os membros do grupo entre si. Com base neste processo de autenticação em grupo deve decorrer posteriormente o processamento para distribuição de chaves em grupo, de acordo com a gestão dinâmica da filiação do grupo.

De acordo com o enquadramento anterior, definem-se as seguintes propriedades fundamentais associadas ao mecanismo de distribuição de chaves: *propriedade de segurança passada*, *propriedade da segurança presente* e *propriedade da segurança futura*.

- A *propriedade de segurança passada* está associada à garantia de confidencialidade da comunicação em grupo numa vista passada. Assim, uma chave distribuída aos membros de um grupo numa dada vista, não pode ser válida no âmbito de uma vista posterior à vista em que foi distribuída. Deste modo, nenhum elemento de uma vista actual do grupo pode pôr em causa a confidencialidade da comunicação em grupo de uma vista anterior, desde que não tenha feito parte de forma lícita dessa mesma vista.

- A propriedade de segurança presente garante que qualquer chave distribuída numa vista anterior do grupo não permite a nenhum dos membros dessa vista pôr em causa a confidencialidade da comunicação em grupo na vista presente, desde que esse membro não faça parte de forma lícita da vista presente.

- A propriedade de segurança futura garante que qualquer chave distribuída em qualquer vista não será válida em vistas futuras do grupo. Assim, nenhum membro de um grupo pode pôr em causa a confidencialidade da comunicação futura em grupo, se não fizer parte de forma lícita da vista futura em que a chave será distribuída.

A estas propriedades podem adicionar-se alguns requisitos específicos, que se passam a introduzir:

- Do ponto de vista das garantias de segurança é muito importante que a geração de chaves durante as várias vistas do grupo seja independente. Esta propriedade adicional, que se pode designar por *independência das chaves de grupo* (ou *group key-independence*) na terminologia mais recente de alguns sistemas da investigação, perspectiva que se um membro fez parte de uma vista de um grupo, não tem por isso quaisquer vantagens acrescidas para efeitos de uma eventual obtenção por via ilícita de um segredo ou nova chave de outra vista do grupo. Esta propriedade coloca logo o problema inicial de se dever evitar que a geração e acordo face a uma nova chave negociada em qualquer vista do grupo se baseie em informação prévia (sejam sementes ou segredos anteriormente partilhados) partilhada por algum ou alguns dos membros resilientes de uma vista anterior do grupo. Esta propriedade, vista como complementar às anteriores leva-nos para uma definição mais exigente das anteriores propriedades. Esta maior exigência tem sido descrita em

trabalhos mais recentes da investigação como uma aproximação correcta às *propriedades de segurança em grupo no futuro e no passado perfeitos*.

O suporte das anteriores propriedades exige que um SCG possua o mecanismo de gestão e distribuição (eventualmente contributiva) das chaves criptográficas baseado num mecanismo de renegociação ou redistribuição dinâmica da chave, sempre que se verifica uma mudança de vista na filiação do grupo. Este aspecto é particularmente agravado pela exigência da propriedade de independência das chaves. O mecanismo de renegociação dinâmica de chaves em grupo face à gestão sincronizada e coerente da vista do grupo é referido na terminologia de língua inglesa como o mecanismo de *rekeying*.

Um dos aspectos mais importantes no estabelecimento de canais seguros para comunicação em grupo prende-se pois com a optimização e eficiência do processamento associado à autenticação em grupo e ao acordo de distribuição (ou negociação) ou redistribuição (ou renegociação) de chaves em grupo. Este aspecto é particularmente relevante devido à eventualidade de ter que se processar o mecanismo de *rekeying* associado à distribuição de chaves de grupo, em contextos de grande probabilidade de entradas e saídas mais ou menos massivas de membros nos grupos bem como cenários em que a ocorrência de falhas que provocam isolamento de membros ou partições de rede pode também ser elevado.

Por outro lado, já anteriormente se referiu que o mecanismo de controlo de acessos em grupo possui características bastante diferentes dos mecanismos de controlo de acessos convencionais utilizados em sistemas distribuídos do tipo cliente/servidor. O controlo de acessos em grupo requer a autenticação em grupo como base para definição de políticas de controlo de acesso ao grupo (e, posteriormente, de operações efectuadas em nome do grupo). No contexto de autenticação em grupo, cada membro do grupo deverá estar habilitado a autenticar todos os membros na vista de um grupo. O controlo de acessos em grupo pode requerer que a autorização de acesso de um membro ao grupo passe pela anuência de todos os membros numa certa vista do grupo (ou eventualmente de uma maioria), o que pode requerer alguma forma de votação ou acordo em grupo sobre a admissibilidade de um novo membro.

Para além das anteriores propriedades associadas à autenticação em grupo, confidencialidade em grupo e controlo de acessos em grupo, um SCG deve permitir que ao nível dos protocolos de comunicação se assegure integridade das mensagens trocadas com base nos protocolos de comunicação em grupo. Esta propriedade é de todas a que menos impacte tem quando comparado o seu suporte no caso de comunicação ponto a ponto. Desde que haja acordo sobre algum algoritmo de síntese de mensagens (ex: MD5, SHA-1, RIPEM) ou eventualmente algum processo híbrido e expedito de assinatura e síntese de mensagens (ex: HMAC) a propriedade pode ser facilmente garantida.

Finalmente, um aspecto importante da integração de suporte de segurança num SCG está relacionado com a forma como os suportes e serviços de segurança podem ser utilizados pelos programadores de aplicações. Como anteriormente se referiu, muitos dos SCG inicialmente apresentados estão hoje disponíveis como *toolkits* e ambientes de programação que fornecem as abstracções e suportes para o desenvolvimento e operação de diferentes aplicações que podem assim beneficiar das propriedades e serviços de base neles contemplados. Também o suporte de serviços de segurança deve ser perspectivado de forma a poder ser facilmente adoptado pelas aplicações.

Uma abordagem possível para este efeito pode basear-se na integração dos serviços de segurança numa camada da pilha protocolar de um SCG, ficando os protocolos acessíveis através das APIs de programação do próprio SCG. Esta alternativa possibilita uma solução de maior transparência em relação às aplicações que adoptam um dado SCG, que poderiam usar os serviços de segurança a partir das mesmas APIs associadas à gestão da filiação dos grupos e dos protocolos de comunicação em grupo. Segundo esta visão as aplicações poderão então usar o mesmo tipo de interfaces de programação de um SCG, adoptem ou não adoptem os serviços de segurança que podem ser configuráveis no momento da instanciação da pilha protocolar a usar. Contudo esta solução pode não ser a mais adequada para requisitos de segurança numa perspectiva extremo-a-extremo. Outra eventual desvantagem é que esta aproximação inviabiliza a possibilidade de partilha de serviços de segurança por parte de diferentes sistemas de comunicação em grupo.

Outra abordagem pode basear-se na interposição de uma camada de segurança entre a aplicação e o SCG, embora integrada com os serviços de base fornecidos pelos SCG. Na maior parte dos casos, as aplicações utilizam os SCG como serviços de *middleware*. Nesta perspectiva os SCG representam um suporte intermédio, imediatamente abaixo dos protocolos de aplicação - que possuem uma perspectiva funcional extremo-a-extremo da comunicação em grupo - e os serviços de comunicação básica processo-a-processo, (suportados de forma nativa nos sistemas operativos, ao nível dos protocolos de transporte, como é o caso da pilha de protocolos TCP/IP). Esta abordagem perspectiva a utilização do suporte de segurança como *serviço de suporte para sessões seguras em grupo*, fornecido com uma camada especializada de *middleware* entre um SCG e as aplicações, com impacte mínimo na sua concepção, desenvolvimento e operação. As aplicações podem ter que ser apenas ligeiramente modificadas quando utilizem ou não utilizem os serviços dessa camada de segurança. Esta solução, não sendo tão transparente do ponto de vista da concretização de serviços de segurança sobre um dado SCG, apresenta porém vantagens interessantes. Os serviços da camada de segurança podem utilizar na base os serviços já fornecidos pelo mecanismo de controlo de filiação dinâmica de processos. Podem também adoptar os protocolos com diferentes semânticas de base de comunicação fiável em grupo que assim podem ser subjacentes aos protocolos de autenticação, acordo de distribuição de chaves e confidencialidade em grupo. Além disso, será possível perspectivar um modelo suficientemente genérico para sessões seguras em grupo que possa vir a ser materializado com base em diferentes SCG. Esta visão é bastante interessante como desafio no sentido da normalização de modelos e serviços de segurança suportados numa perspectiva multi-plataforma e num contexto de independência da tecnologia de cada SCG em particular.

1.6. Objectivo e contribuições da dissertação

A presente dissertação focaliza-se na problemática de adicionar e integrar serviços de segurança orientada para grupos em sistemas de comunicação fiável em grupo. O objectivo é o estudo e a caracterização do suporte específico para autenticação em grupo, controlo de acessos em grupo e confidencialidade em grupo. Com base nessa análise e a partir do suporte genérico de um sistema de comunicação em grupo, propõe-se um modelo que integra um serviço de autenticação, um sistema de autorizações com mecanismo

de controlo de acessos e um protocolo de gestão e distribuição de chaves criptográficas para sessões de comunicação em grupo.

A integração destes serviços faz-se sobre uma camada de segurança (SGL - *Secure Group Layer*) capaz de fornecer uma abstracção de segurança integral para desenvolvimento de aplicações baseadas em interacções em grupo. A perspectiva de estruturação desta camada vai no sentido de uma aproximação baseada no estabelecimento de um serviço de *sessão segura de comunicação fiável em grupo* que possa ser eventualmente implementável com base em diferentes plataformas de SCG, desde que disponibilizem um determinado conjunto de abstracções de base de programação.

1.7. Organização da dissertação

A dissertação está organizada nos seguintes capítulos do seguinte modo: o capítulo 2 apresenta os fundamentos e aspectos essenciais associados à conceptualização, terminologia e enquadramento dos sistemas de comunicação em grupo (SCG). Este capítulo descreve as características e propriedades principais proporcionadas pelos SCG para suporte de desenvolvimento e operação de aplicações baseadas em interacções de grupos de processos. No capítulo 3 introduzem-se alguns princípios, propriedades e serviços de segurança na perspectiva de um sistema distribuído. Estes princípios, propriedades e serviços são essencialmente apresentados do ponto de vista da sua conceptualização, estruturação e suporte tendo em conta o enquadramento e a relevância para a perspectiva da dissertação.

O capítulo 4 apresenta uma síntese sobre os aspectos fundamentais associados à satisfação de propriedades de segurança em Sistemas de Comunicação em Grupo. A aproximação recente de alguns trabalhos em relação à integração de serviços de segurança em sistemas de comunicação em grupo é apresentada como trabalho relacionado com os objectivos e contribuições da dissertação. O capítulo 5 apresenta a proposta de um modelo de integração de um serviço de segurança para um sistema de comunicação fiável em grupo. Este serviço é composto por vários subsistemas, a saber: sistema de autenticação em grupo, sistema de autorização e mecanismo de controlo de acessos em grupo e sistema de geração e distribuição de chaves de sessão para comunicação confidencial em grupo. No capítulo 5 formaliza-se as conclusões e futuro trabalho a ser pesquisado e desenvolvido.

Capitulo 2 - Sistemas de Comunicação em Grupo

Neste capítulo apresentam-se alguns fundamentos e aspectos essenciais associados à conceptualização, terminologia e enquadramento dos sistemas de comunicação em grupo (SCG). Descrevem-se as características e propriedades principais que esses sistemas proporcionam para suporte de desenvolvimento e operação de aplicações baseadas em interacções de grupos de processos. Os aspectos introduzidos referem-se sobretudo às questões essenciais relativas ao suporte de base que aqueles sistemas disponibilizam tendo em vista a perspectiva e os objectivos da presente dissertação.

2.1. Fundamentos dos sistemas de comunicação em grupo

Os sistemas para comunicação em grupo têm como objectivo proporcionar um suporte de comunicação fiável entre processos filiados e geridos de forma integrada em grupos de processos. Os processos que integram um grupo podem estar logicamente ou fisicamente distribuídos no âmbito de uma rede de computação de menor ou maior escala.

Com a diversidade dos tipos de comunicação existentes actualmente, os sistemas de comunicação em grupo são resistente a falhas, tais como atrasos nas comunicações, trocas na ordem das mensagens e partições de rede.

O principal problema da comunicação em grupo consiste na relação número de membros e sua eventual distribuição (vertente de escalabilidade do suporte) e o desempenho dos protocolos de comunicação e gestão da organização ou filiação do grupo (vertente associada à eficiência do suporte de sistema). A materialização da abstracção da entidade “grupo de processos” como entidade principal do suporte de um sistema de comunicação em grupo, as propriedades de fiabilidade associadas às diferentes semânticas de comunicação em grupo do ponto de vista de diversos modelos de tolerância a falhas ao nível dos processos, dos processadores e do meio de comunicação que utilizam e as características de escalabilidade e eficiência do suporte de protocolos que proporcionam, estiveram na génese da investigação, concepção e disponibilização de diversos sistemas para comunicação fiável em grupo a partir da investigação desenvolvida ao longo da última década do século XX.

2.2. Caracterização e propriedades de um SCG

Os sistemas de SCG fornecem uma camada transparente para gestão e disseminação de dados de um grupo.

Um SCG deve suportar cinco eventos básicos:

- *Join* – Adição de um novo membro a um grupo
- *Leave* – Saída de um membro do grupo.
- *Partition* – Partição no sistema de comunicação impedindo a comunicação com um ou mais membros do grupo.
- *Merge* – Reunificação do grupo após uma partição.
- *Send* – Envio de dados por parte de um membro para o grupo.

A gestão do grupo permite detectar falhas e facilita a recuperação das mesmas. Esta gestão é conseguida através do conceito de *membership* que está presente nas semânticas de *Extended Virtual Synchrony* [1] e *Virtual Synchrony* [2,3].

A disseminação dos dados é critica já que é necessário que todos os membros do grupo os recebam sempre, e muitas aplicações exigem que os mesmos sejam entregues de forma ordenada. Um sistema de comunicação fiável em grupo torna-se imprescindível para um SCG.

2.3. Protocolos de comunicação fiável em grupo

Um sistema de comunicação fiável em grupo é complexo e o seu desempenho depende do meio físico de comunicação.

O problema da comunicação fiável em grupo consiste na necessidade de escalabilidade, tendo de suportar fluxos dinâmicos de entrada e saída de membros. Contudo a comunicação fiável em grupo não depende só de garantir a entrega dos dados, já que existem aplicações que necessitam que estes sejam também entregues de forma ordenada.

Dos protocolos mais conhecidos para comunicações ponto a ponto, o TCP/IP parece ser um candidato a ser utilizado. No entanto este protocolo apresenta problemas ao ser estendido para suportar grupos. As suas propriedades de fiabilidade requerem um circuito virtual estabelecido entre as duas partes, *acknowledgments* por parte do receptor e retransmissão, que é sempre mantida pelo emissor.

Num grupo com comunicações do tipo *one-to-many* ou *many-to-many* não é viável ter-se *acknowledgements* para todos os pacotes recebidos, nem retransmissões contínuas dos mesmos quando não chegam ao destino.

O outro protocolo utilizado, UDP, é um protocolo composto por datagramas e que não suporta, na sua forma básica, qualquer tipo de fiabilidade.

Dado que a sua utilização implica um menor custo de gestão e menores atrasos na comunicação, uma alternativa para comunicação em grupo baseia-se em extensões a este protocolo, como é o caso do IP *Multicast*.

Finalmente, uma das soluções para comunicação fiável em grupo consiste em estender o IP *Multicast* de forma a suportar fiabilidade na entrega dos dados e a ordenação dos mesmos. Como é, por exemplo, o caso do RMP [37] e do MTP [38].

2.4. Semânticas de comunicação fiável em grupo

O uso de protocolos fiáveis não é suficiente para garantir a entrega fiável das mensagens a todos os membros do grupo. A vertente da ordenação das mesmas é essencial para garantir a entrega de forma correcta das mensagens. Foram desenvolvidos algoritmos para lidar com esta problemática, em que se destaca os algoritmos que garantem a ordem FIFO (*First In First Out*), Causal, Total e *Safe*.

A ordem FIFO garante que duas mensagens enviadas pelo mesmo emissor sejam recebidas pela mesma ordem de envio.

A ordem Causal, desenvolvido por Lamport [4] é descreve que uma mensagem m_1 procede m_2, designando-se por $m_1 \rightarrow m_2$, se e apenas se:

- m_1 e m_2 foram emitidas pelo mesmo processo e m_2 foi enviada após m_1 ou;
- m_1 foi entregue ao emissor de m_2 antes de m_2 ter sido emitida ou;
- Existe m_3 tal que $m_1 \rightarrow m_3$ e $m_3 \rightarrow m_2$.

A ordem Total garante que duas mensagens entregues a um par de processos são entregues pela mesma ordem a ambos.

A ordem *Safe* garante que as mensagens apenas são entregues aos destinatários finais após ter havido um *acknowledge* por parte de todos os membros.

Estas semânticas, juntamente com a garantia da entrega das mensagens por parte dos protocolos, garantem a fiabilidade necessária para o estabelecimento dos canais de comunicação dos SCG. Porém não se pode definir um SCG apenas pelos seus canais de comunicação, a gestão da filiação em grupo é de extrema importância já que define os membros actuais de um grupo. Em seguida apresenta-se este conceito.

2.5. Gestão de filiação em grupo

Devido à natureza dinâmica da entrada e saída de membros do grupo, e à possibilidade de ocorrerem falhas, torna-se necessária a existência da gestão do estado do grupo.

Um SCG utiliza o conceito de *membership* para a gestão da filiação em grupos. Este conceito define o estado actual dos membros de um grupo, ou seja, mantém registo dos membros activos e inactivos em cada grupo, podendo desta forma construir para cada um dos grupos a árvore de membros. Estas árvores são essenciais para que seja garantida a disseminação fiável dos dados e servem como forma de disseminação de entrada/saída de membros.

A gestão do grupo oferece alta disponibilidade, sendo resistente às falhas dos membros, partições de redes e suportando recuperações de membros através das reunificações de redes.

O estado de um grupo pode sofrer as seguintes alterações:

- Entrada de membros no grupo.
- Saída de membros no grupo.
- Partições de redes (podem ser consideradas como saídas de membros).
- Reunificação de redes (podem ser consideradas como entradas de membros).

Sempre que há uma alteração ao estado do grupo esta é propagada para todos os membros, e é actualizada a respectiva árvore.

Como já foi referido, este conceito é utilizado em conjunto com sistemas que garantem a sincronia do estado do grupo em todos os membros. O seu uso é descrito em 2.7.

A necessidade da garantia da entrega dos dados, pode não satisfazer todos os requisitos de uma aplicação, a ordenação dos mesmos é, também, um dos aspectos importantes que um SCG fornece. Em seguida iremos abordar esta questão.

2.6. Problemática da ordenação de mensagens

Foram apresentados algoritmos de ordenação de mensagens, no entanto a problemática da sua integração em sistemas de grupo é de relevo e apresenta-se técnicas existentes para a sua implementação em SCG.

A problemática pode-se reduzir à *performance* dos algoritmos existentes em redes com latências dinâmicas e com um número considerável de membros. É com estes problemas em mente que se apresenta a implementação dos algoritmos de ordenação apresentados anteriormente.

A ordem FIFO pode ser implementada através da simples marcação das mensagens do emissor com um número de sequência. A entrega destas mensagens é feita com base nesse número de sequência.

A ordem Causal é implementada com relógios lógicos em casos de sistemas assíncronos ou com estampilhas com o tempo real para sistemas síncronos. Para o uso de SCG consideram-se sempre os sistemas assíncronos.

A implementação da ordem Total em sistemas assíncronos consiste em algumas alternativas, a primeira hipótese consiste na negociação por parte dos destinatários do número de ordem para cada mensagem recebida. Outra solução consiste em delegar a um processo a responsabilidade de atribuir uma ordenação para as mensagens. Desta forma este processo torna-se um sequenciador de mensagens. Esta solução é a mais utilizada pelos SCG já que toda a informação tem que ser transmitida ao SCG para difusão para o grupo. Ainda existe uma outra solução que consiste numa extensão dos relógios lógicos para relógios vectoriais e em que as mensagens são recebidas pela sua ordem parcial e mensagens concorrentes são totalmente ordenadas utilizando uma função determinística.

A ordem *Safe* é uma extensão da ordem Total onde se tem um *buffer* para guardar as mensagens pendentes de entrega, só depois do *acknowledge* da recepção da mensagem pelo *buffer* de todos os membros, é que esta é entregue a cada membro de forma definitiva.

2.7. Suporte para sincronismo virtual

A Sincronia Virtual (*Virtual Synchrony – VS)* permite garantir que todas as alterações ao estado de um grupo são observadas da mesma forma por todos os membros do grupo.

Estas alterações de *membership* são totalmente ordenadas em relação às mensagens "regulares" que são transmitidas por membros dos grupos. O estado actual do grupo designa-se por *view;* todos os membros do grupo têm conhecimento da mesma *view* no mesmo instante.

O *VS* implementa a gestão da *membership* através da propriedade *failure atomicity*. Esta propriedade consiste na entrega das mensagens recebidas entre *views* diferentes do grupo. Estas mensagens ou são entregues a todos os membros, ou não são entregues a nenhum.

O problema deste protocolo de gestão de *membership* reside no facto de não garantir a entrega e ordenação de dados durante uma alteração à *view* actual do grupo.

Com o objectivo de garantir o problema da entrega e ordenação de dados durante as alterações às *views,* o protocolo de sincronia virtual foi estendido, tendo sido criada a sincronia virtual estendida (*Extended Virtual Synchrony – EVS*).

Da extensão resultou uma nova fase entre as *views* em que a nova *view* ainda não foi aceite por todos os membros mas a ordenação e entrega das mensagens é garantida durante a transacção de *views*. Essa fase é chamada configuração transaccional do grupo, e tem como função garantir que os membros que estavam numa configuração estável e que irão permanecer na nova configuração do grupo tenham a certeza de que as mensagens enviadas durante a transacção do estado do grupo foram recebidas ordenadamente.

Capitulo 3 – Fundamentos de segurança em sistemas distribuídos

Neste capítulo introduzem-se alguns princípios, propriedades e serviços de segurança na perspectiva de um sistema distribuído. Estes princípios, propriedades e serviços são essencialmente apresentados do ponto de vista da sua conceptualização, estruturação e suporte tendo em conta o enquadramento e a relevância para o âmbito da dissertação.

3.1. Propriedades de segurança num sistema distribuído

A segurança num sistema distribuído tem como base o conceito de segurança informática entre entidades. No entanto, a definição de segurança informática não é trivial. A sua definição mais comum é através da descrição das suas propriedades:

- Confidencialidade – assegura que apenas entidades autorizadas possam aceder à informação.
- Autenticidade – assegura a identidade dos intervenientes.
- Disponibilidade – assegura a operacionalidade e funcionalidade continua.
- Integridade – assegura que a alteração ilegítima de uma mensagem seja detectada.
- Não Repudio – assegura que os emissores não podem negar a emissão de uma mensagem.

A obtenção destas propriedades numa mensagem obriga ao uso de diferentes técnicas de criptografia.

Para a obtenção da disponibilidade normalmente utiliza-se redundância, como por exemplo o uso de servidores de DNS secundários.

A integridade implica o uso de um *message digest* [5,6], que permite calcular uma "marca de água".

O não repudio está intrinsecamente relacionado com a autenticidade, e é garantido através do uso de assinaturas digitais.

Em seguida veremos com mais pormenor as propriedades: confidencialidade, autenticidade e integridade.

3.1.1. Confidencialidade

Como já foi referido, a confidencialidade garante o acesso à informação apenas pelas pessoas autorizadas. O seu ponto forte passa por provocar entropia na informação, de tal modo que só pessoas autorizadas consigam obter a informação original.

Devido à sua necessidade em acções militares, este método existe já há muito tempo mas tem sofrido uma lenta evolução.

A confidencialidade é obtida através do uso de criptografia. Actualmente existem dois ramos da criptografia - simétrica e assimétrica.

3.1.1.1. Criptografia simétrica

A criptografia simétrica é um dos métodos usados para garantir a confidencialidade. Consiste na utilização de uma única chave para cifrar e decifrar uma mensagem. Todos os intervenientes têm que partilhar essa chave.

O problema deste tipo de criptografia é a necessidade de confiança nos intervenientes, pois caso um dos participantes divulgue a chave, ou esta seja descoberta, será necessário renegociar a chave e distribui-la.

Este método é rápido a cifrar e a decifrar mensagens. As chaves utilizadas têm um tamanho relativamente pequeno (entre os 56 e 256 bits) e o seu uso é muito divulgado.

Actualmente os algoritmos de cifra simétrica mais divulgados são o DES [7,8], 3DES [11], AES [12], IDEA [9] e Blowfish [10].

3.1.1.2. Criptografia assimétrica

Com o objectivo de suprimir a necessidade de relações de confiança nos intervenientes, foi desenvolvida a criptografia assimétrica. Este método consiste na utilização de duas chaves:

- Uma chave privada apenas conhecida pelo utilizador que a criou
- Uma chave publica, que é divulgada livremente.

Devido ao uso de uma chave pública, este método também é conhecido por *public-key cryptography*.

Para cifrar utiliza-se a chave pública, sendo necessário a utilização da chave privada para se obter a informação original.

Este método também é utilizado para assinaturas digitais, onde a chave privada é utilizada para cifrar o *digest* da mensagem, e a chave pública para o decifrar.

O seu custo de utilização é elevado devido ao tamanho da chave pública, que pode variar entre os 512 (em desuso) e os 4096 bits. A utilização mais frequente deste método na vertente de cifra é para se estabelecer um canal seguro, de forma a ser possível negociar uma chave simétrica e prosseguir com criptografia simétrica. Um exemplo de cifra assimétrica é o RSA [13].

Para ambos os modelos o processo de geração de chaves é facilmente perceptível, quer seja apenas uma (cifra simétrica), quer seja um par de chaves (cifra assimétrica), assim como a distribuição das mesmas através de canais inseguros.

3.1.2. Geração e distribuição de chaves criptográficas

A geração e distribuição de chaves é uma problemática que existiu desde o início da criptografia. A problemática da confiança das entidades no uso de cifra simétrica, resume-se à questão da descoberta ou divulgação da chave de cifra. Neste caso seria necessário gerar uma nova chave e, mais importante ainda, distribui-la de forma segura. A geração da chave deve corresponder a certos requisitos matemáticos, e para a sua distribuição deverá sempre ser assumido um canal inseguro.

A primeira solução adoptada foi confiar numa TCB, que gerava e distribuía chaves.

3.1.2.1. Modelo de Needham Schroeder com chaves simétricas

Este modelo é baseado numa relação de confiança sobre uma entidade externa, que gera e distribui uma chave de sessão para ser utilizada entre dois principais. Este modelo, chamado *Needham-Schroeder Symetric-Key Protocol* [14], distribui chaves simétricas.

No modelo base temos três principais: o principal A, que deseja conversar de forma segura com o principal B; e o servidor que gera e distribui as chaves (*Key Distribution Center* - KDC).

O KDC tem chaves secretas, negociadas antes do protocolo, para cada um dos principais A e B. O protocolo utiliza números aleatórios, denominados *nonce,* com o objectivo de proteger o protocolo contra ataques de *replay*.

3.1.2.2. Variantes do protocolo Needham Schroeder

A segurança do protocolo descrito acima pode ser quebrada com o uso de *replay* durante a fase de autenticação dos principais. Este ataque foi descoberto por Denning e Sacco [15] e pressupõe que o malfeitor, denominado *Mallory* (M), gravou a fase de distribuição da chave de sessão, conseguindo portanto obtê-la.

A solução proposta por Denning e Saco para evitar este ataque consiste na adição de estampilhas temporais (*timestamps*) nas mensagens.

Como já foi referido estes protocolos pressupõem um TCB que tenha chaves de cifra para os intervenientes. Em muitas aplicações este pressuposto não é viável, tendo sido criado um outro protocolo que permite negociar uma chave simétrica a partir de um canal inseguro.

3.1.3. Modelo de Diffie-Hellman

O modelo de Diffie-Hellman [16] utiliza um canal inseguro e as propriedades matemáticas da exponenciação para negociar uma chave simétrica.

O protocolo utiliza dois parâmetros públicos (g e p). O parâmetro p consiste num número primo e o parâmetro g é um numero gerador inferior a p tal que para todo o número entre 1 e p-1 existe uma potência k que satisfaz: $n = g^k \bmod p$.

Os intervenientes geram números a e $b \in \{1,...,p-2\}$ e computam os seus valores públicos utilizando os parâmetros públicos g e p. O valor público dos intervenientes é $g^a \bmod p$ e $g^b \bmod p$, Trocam estes valores públicos pelo canal inseguro e computam $g^{ab} = (g^b)^a \bmod p$ e $g^{ba} = (g^a)^b \bmod p$. Devido às propriedades da exponenciação $g^{ab} = g^{ba} = k$, sendo esta a chave secreta gerada e acordada pelos intervenientes.

Devido à falta de autenticação dos principais, este protocolo é vulnerável ao ataque de *Man-in-the-Middle* (*MiM*). Este ataque consiste no processo em que a troca dos valores é interceptada por uma terceira identidade situada entre os intervenientes, ficando esta identidade a fazer o "routing" das mensagens entre os dois principais.

Para evitar este ataque é necessário assinar as mensagens que contêm os valores públicos trocados entre os principais.

3.1.4. Distribuição de chaves com assinaturas RSA

Esta distribuição já foi referida como sendo mais utilizada pela criptografia assimétrica de forma a autentificar os intervenientes e gerar uma chave de sessão, tipicamente do tipo simétrica. Utiliza o conceito de certificado digital que consiste em três campos essenciais:

- Uma chave pública.
- Informação relacionada (Identidade do utilizador, tal como nome, identificador, etc.).
- Uma ou mais assinaturas digitais.

Mais adiante será abordado com mais pormenor o uso e validação de certificados digitais.

Este modelo de autenticação e distribuição tornou-se popular com a sua utilização no protocolo SSL [18]. Este modelo pode ser estendido para que os intervenientes apresentem mutuamente certificados e se autentiquem, podendo ser feito um controlo de acessos ao nível do utilizador.

No próximo capítulo analisa-se técnicas de autenticação que são essenciais para garantir a integridade dos protocolos de controlo de acessos.

Com este protocolo concluímos uma abordagem a algumas técnicas de referência para o princípio de segurança de confidencialidade. Em seguida iremos abordar técnicas que lidam com o princípio de autenticidade e controlo de acessos.

3.1.5. Autenticação de principais

A autenticidade de principais tem como objectivo garantir a veracidade da identificação apresentada pelo principal. Por exemplo, o Bilhete de Identidade tem essa função, ao assegurar a identificação de um indivíduo.

Ao longo do tempo foram desenvolvidas técnicas criptográficas que permitem assegurar a identidade de uma mensagem ou pessoa através de duas maneiras: cifras simétricas e assinaturas digitais.

É discutível a validade da cifra simétrica como forma de autenticar um principal, já que esta forma de autenticação é baseada no facto de que apenas as partes interessadas têm conhecimento do segredo utilizado para a cifra. O principal ponto de falha desta forma de autenticação consiste na possibilidade de um dos participantes divulgar o segredo a um terceiro, ou no facto do segredo poder ser descoberto por um terceiro.

As assinaturas digitais são a forma mais divulgada de autenticidade, pois não é necessário confiar na outra parte para validar a veracidade da assinatura. Apesar de não haver necessidade de confiança entre os interlocutores, é no entanto necessária uma *Trusted Computing Base*(TCB), uma identidade externa aos intervenientes que assegura a validade e integridade das suas identidades.

A assinatura digital de uma mensagem consiste num *Message Digest* da mensagem, que é cifrado em seguida. Para efectuar a sua validação, o receptor faz por sua vez o *digest* da mensagem recebida, decifra a assinatura e verifica se os *digests* são iguais.

Para evitar a negociação do segredo da validação das assinaturas utiliza-se cifra assimétrica.

3.1.6. Integridade dos dados

A propriedade de integridade proporciona a garantia da detecção de alterações efectuadas aos dados. O seu aparecimento deve-se em grande parte à falta de confiança na transmissão e nos sistemas de salvaguarda de dados, onde os ruídos no canal de comunicação podiam alterar os dados durante a sua transmissão entre o emissor e receptor, e onde os sistemas magnéticos para salvaguarda de informação eram propícios a erros. Para combater este problema surgiu um sistema que detecta a alteração da informação. Este sistema inicial foi designado por CRC (*Cyclic Redundacy Check*), e a sua base matemática reside na divisão binária e no uso do resto dessa divisão para cálculo do CRC.

Porém, este sistema simples pode não detectar todas as alterações efectuadas a dados. A necessidade de garantir a integridade dos dados de forma mais fiável levou ao desenvolvimento de funções de *hash*. Estas

funções garantem que, seja H a função de *hash*, *x* e *y* valores de entrada para a função, então se $H(x) \neq H(y)$ implica que $x \neq y$ e caso $H(x) = H(y)$ então $x = y$. Actualmente existem duas importantes implementações criptográficas deste tipo de funções, o MD5 e o SHA-1 que serão abordadas em pormenor mais adiante.

3.1.6.1. Algoritmos de síntese de mensagens

A síntese de uma mensagem consiste na utilização de funções de *hash* criptográficas para a criação de um número único. Estas funções de *hash* são consideradas seguras desde que não seja possível em tempo fazível a computação da mensagem original a partir do *hash* e caso não existam colisões. Uma colisão sucede quando duas mensagens diferentes resultam no mesmo resultado de *hash*.

Um dos algoritmos mais utilizados na síntese de mensagens é o MD5 [36]. Este algoritmo foi o sucessor do MD4 após ter sido demonstrado a insegurança deste último. Produz marcas de água de 128 bits e actualmente o MD5 é muito utilizado em distribuição de software como forma de prevenir a alteração do código e evitar a instalação de programas com código malicioso incluído. Este algoritmo foi muito utilizado até ser ter descoberto algumas falhas no mesmo, pois é possível obter-se a mesma *hash* com pares de mensagens diferentes. Como alternativa surgiu o SHA-1 [47] que produz *digests* de 160 bits, o que torna o algoritmo mais seguro contra ataques de colisão e mais difícil a inversão do *hash* para se obter a mensagem original.

Estes algoritmos assumiram um papel muito importante na utilização de assinaturas digitais. Estas assinaturas consistem na utilização de uma função de síntese de mensagens para se obter uma marca de água e depois cifrar essa marca de água de forma a garantir que a mensagem foi enviada pela entidade respectiva e que não foi alterada durante o seu percurso. Como exemplo temos a variante HMAC que consiste no uso do MD5 ou SHA-1 e de cifrar o resultado desse *hash* com cifra simétrica.

3.1.7. Problemática de controlo de acessos

O controlo de acessos tem como objectivo garantir que identidades diferentes tenham diferentes permissões para aceder a recursos.

O controlo de acessos é fiável desde que o protocolo de autenticação das entidades e as políticas de acessos sejam também fiáveis.

Nos sistemas distribuídos actuais, em que utilizadores desejam utilizar recursos localizados em servidores, deseja-se que os servidores que alojam esses recursos restrinjam o seu acesso apenas a utilizadores autorizados e autenticados. Neste ambiente existem vários riscos:

- Um utilizador pode aceder a um recurso fazendo-se passar por outro utilizador.
- Um utilizador pode personificar a sua máquina por uma outra máquina.
- Um utilizador pode recolher as mensagens de autenticação, reenvia-las de forma a iludir o processo de autenticação e obter acessos indevidos

Com o objectivo de resolver estes problemas de segurança apareceram vários protocolos, entre os quais o Kerberos [17].

3.1.7.1. Sistema Kerberos

O Kerberos fornece um serviço centralizado de autenticação e controlo de acessos cuja função é autenticar utilizadores perante servidores, e servidores perante utilizadores.

O Kerberos utiliza criptografia convencional (cifra simétrica - DES), *timestamps* e *ticket*s. O seu princípio baseia-se no KDC do modelo de Needham-Schroeder.

Este modelo utiliza dois componentes independentes: o *Authentication Server* (AS) e o *Ticket Granting Server* (TGS).

O AS é o componente que conhece todas as *passwords* e que as guarda numa base de dados centralizada. Este componente também contém chaves secretas de cifra para cada servidor e é responsável pela autenticação dos utilizadores.

O TGS é o componente responsável pela geração, manutenção e distribuição de *tickets* aos utilizadores autenticados pelo AS. É também o componente responsável pelo controlo dos acessos dos utilizadores aos servidores. Estes *tickets* são apresentados aos servidores para comprovar a validade das permissões dos utilizadores.

Nas mensagens finais do protocolo é feita uma autenticação mútua entre o servidor e o utilizador.

Este sistema é muito divulgado e utilizado, por exemplo no sistema operativo Windows 2000 Server.

3.1.7.2. Autenticação baseada em autoridades de certificação

Dos protocolos vistos anteriormente e na utilização da cifra assimétrica é preciso ter atenção à gestão e verificação da validade dos certificados digitais. Existem algumas aproximações que contribuíram para este tema, e que serão analisadas a seguir.

Este modelo é muito divulgado e o seu aparecimento está associado à cifra assimétrica. O principal objectivo deste modelo é validar certificados digitais garantindo que cada certificado pertence a uma entidade, assegurando-se também da identidade da mesma. Para se alcançar este objectivo tem-se uma Autoridade de Certificação (Certificate Authority – CA) consistindo numa entidade de administração central, que assina e publica certificados digitais, bem como a respectiva chave pública.

Para evitar certificados falsos a chave pública da CA tem que ser fiável, logo a CA ou publica a sua chave ou tem a sua chave assinada por outra CA. Por exemplo a chave do CA pode ser publicada no jornal ou, no caso mais comum, vem já embutida em *browsers*. Com a assinatura de uma chave pública de uma CA por outra CA criou-se o conceito de hierarquias de CAs, onde por exemplo podemos ter uma CA central que apenas publica chaves de sub-CAs, e ter-se uma CA para publicar chaves para comércio electrónico, escolas, indivíduos, empresas, etc.

A emissão de um certificado efectua-se da seguinte forma:

- Alice gera o seu par de chaves e envia a chave pública para o CA, bem como alguma forma de identificação.
- O CA verifica a identificação e efectua mais alguns passos, opcionais, para garantir a identidade de Alice e que a chave publica não foi modificada durante o envio.
- O CA envia um certificado confirmando a validade da chave pública de Alice assinado por si. Caso estejamos perante uma hierarquia de CAs o certificado conterá toda a cadeia de assinaturas dos CAs.

A escalabilidade deste modelo só é possível devido ao facto de os certificados digitais terem um formato *standard*, conhecido como X.509 [19], que especifica os campos de um certificado digital, e que permite a interoperabilidade dos vários emissores de certificados.

É de notar que todo este modelo depende do CA e como tal um ataque ao CA é o seu principal ponto de falha. Uma chave comprometida de um CA levanta graves problemas já que todo o certificado digital publicado por este CA é passível de ser falsificado. Adicionalmente, pode ser necessário revogar um certificado digital antes de expirar o seu tempo de validade. Para suprir esta necessidade, surgiu o *Certification Revocation List* (CRL), que consiste numa lista de certificados não válidos e que deve ser consultada sempre que for necessário verificar a validade de um certificado.

Como já foi referido, o principal problema deste modelo consiste na confiança numa entidade central. Para ultrapassar este problema outros modelos foram apresentados.

3.1.7.3. Autenticação baseada em redes de confiança

Este modelo utilizado em aplicações do tipo *Peer to Peer* (P2P) tem como impulsionador o PGP [21].

A grande diferença entre este modelo e o visto no ponto anterior consiste na validação da confiança que se tem num certificado digital.

No caso anterior existem entidades específicas que validam o certificado digital, neste modelo os certificados digitais podem ser validados por outras pessoas.

Desta forma existem três distintos tipos de validade/confiança nos certificados digitais:

- Directo
- Hierárquico
- *Web of Trust*

No primeiro tipo, um indivíduo acredita no certificado apresentado por outro porque sabe a origem do mesmo. Tipicamente este modelo é usado entre amigos e família que se conhecem e se acreditam mutuamente. Este tipo de validade/confiança de certificados pode ser mapeado para o modelo baseado em autoridades de certificação, em que se acredita num CA.

No segundo caso temos entidades terceiras em cuja integridade acreditamos e que divulgam certificados digitais. Este tipo também pode ser mapeado para o modelo anterior, no caso em que se utiliza as hierarquias de CAs.

O último tipo de validade/confiança de certificados não tem mapeamento para o modelo de CAs. Baseia-se no conceito de que qualquer pessoa no mundo consegue encontrar uma ligação com qualquer outra pessoa no mundo, utilizando seis ou menos intermediários. Neste caso um individuo assina um certificado digital como sendo válido, tornando-se um porta-voz daquele certificado. Ao estender este processo obtém-se uma "teia de confiança" entre os utilizadores.

Com este último tipo um indivíduo apenas acredita nos certificados assinados por outro indivíduo se este for credível, de resto não têm qualquer validade.

Como se pode entender, este modelo elimina a necessidade de entidades terceiras de administração publicarem a sua chave pública, ou de virem já embutidas em aplicações, transferindo a opção de confiança num determinado certificado para os utilizadores.

3.2. Problemática dos sistemas distribuídos

Já foi referido que a segurança em sistemas distribuídos tem como a base o conceito de segurança base. Neste sub capitulo iremos aprofundar a aproximação de segurança em sistemas distribuídos.

A extensão de confidencialidade para sistemas distribuídos é complexa, não por causa dos protocolos de cifra a utilizar, mas sim devido à geração e distribuição da cifra a utilizar. A problemática da geração da cifra deve-se ao facto de se ter duas formas possíveis para a geração da mesma: centralizada e contributiva. A problemática da distribuição resume-se a disseminar a chave de cifra de forma segura e fiável, utilizando o menor número de mensagens possível. Estas duas problemáticas são apresentadas e discutidas em detalhe no próximo capitulo.

A autenticação de principais neste tipo de sistemas está influenciada com a escalabilidade do tipo de autenticação. Em primeiro lugar deve-se separar dois grupos de autenticação: dos principais e mensagens dos principais.

A autenticação dos principais é obtida, normalmente, com o recursos de uma entidade externa (à lá Kerberos) que valida a veracidade das credenciais apresentadas pelo principal.

A autenticação das mensagens enviadas pelos principais num sistema distribuído é obtida, ou pelo uso de assinaturas digitais singulares, ou pelo uso de assinaturas digitais de grupo, ou ainda através de uma HMAC partilhada pelo grupo. Actualmente o uso de assinaturas digitais são restritivas, devido à falta de escalabilidade no sistema de assinaturas digitais pessoais, e devido ao estado ainda precoce do desenvolvimento cientifico das assinaturas digitais de grupo.

No próximo capitulo são discutidas as várias opções a utilizar num sistema de comunicação em grupo para garantir as propriedades da segurança.

A disponibilidade de um sistema distribuído depara-se com duas questões fundamentais: a garantia da

existência dos serviços em todos os principais e da consistência desses mesmos serviços. Os sistemas de comunicação fiável e resistentes a falhas são apresentados no próximo capitulo como forma de garantir disponibilidade.

A integridade no âmbito de sistemas distribuídos é conseguida através dos mesmos métodos utilizados em comunicações ponto a ponto.

O não repúdio está directamente relacionado com a autenticação das mensagens trocadas entre os principais. Como tal a sua analise no âmbito de sistemas distribuídos é remetida para a analise da autenticação efectuado no próximo capitulo.

Capitulo 4 - Segurança em SCG

Neste capítulo apresenta-se uma síntese sobre os aspectos fundamentais associados à satisfação de propriedades de segurança em Sistemas de Comunicação em Grupo. A aproximação recente de alguns trabalhos em relação à integração de serviços de segurança em sistemas de comunicação em grupo é apresentada como trabalho relacionado com os objectivos e contribuições da dissertação.

4.1. Geração e distribuição de chaves criptográficas

Já foi abordado a problemática da geração e distribuição de chaves para indivíduos, de seguida iremos abordar esta mesma problemática mas para o conceito de grupos. Quando nos deparamos com um grupo, temos de ter em conta que não se pode assumir que os seus membros são estáticos, isto é, os membros de um grupo entram e saiem do grupo voluntáriamente e, possivelmente, sem avisar o grupo da sua proxima acção. Devido à característica dinâmica de um grupo, é natural desejar-se que uma alteração ao estado do grupo ou a expiração de um temporizador provoque a geração e distribuição de uma nova chave de grupo.

A chave de um grupo está sujeita a politicas de segurança que devem ser garantidas pela gestão da chave. Esta gestão tem como objectivos garantir:

- Autentificação e identificação dos membros do grupo – garante que não seja permitido a intrusos entrar num grupo e que não seja possivel a um impostor assumir a identidade de outrém.
- Controlo de acessos - verificar se um membro tem privilégios suficientes para receber a chave de grupo e efectuar operações no grupo.
- Geração, distribuição e certificação da instalação de chaves de grupo – Poderá ser necessário alterar a chave de grupo a intervalos regulares e garantir a independência da nova chave em relação à anterior.

Nos próximos sub capítulos iremos analisar implementações dos vários tipos de geração e distribuição de chaves: centralizada, distribuida e mista.

4.1.1. Geração e distribuição de chaves centralizada

O protocolo mais simples e básico de geração e distribuição de chaves de forma centralizada resume-se a um KDC que tem tuplos do tipo: <membro, chave de distribuição> para cada membro que exista num grupo. Neste modelo o KDC gera a chave sem auxilio externo e distribui-a por todos os membros, ponto a ponto, utilizando uma chave de distribuição. Esta operação envolve a geração e disseminação de *n* mensagens para os *n* membros do grupo. Para grupos com um grande número de membros esta aproximação torna-se difícil de gerir, com pouca escalabilidade e um elevado custo do envio da chave. Note-se que o custo de envio da nova chave é composto pelo custo de cifrar a nova chave de grupo com a chave de distribuição acrescido do custo de enviar a mensagem pela rede para os membros.

Nesta classe de protocolos o seu desempenho é medido por quatro factores:

- Requisitos de armazenamento – O número de *key encryption keys* (KEKs) que os membros e o KDC precisam de guardar.
- Tamanho das mensagens – O tamanho da mensagem de refrescamento da chave para os eventos de *join* e remoção de um membro. O envio pode ser efectuado através de *multicast* ou *unicast*, tendo este último um maior custo devido à necessidade de estabelecimento de um canal seguro.
- Segurança passada e segurança futura – Duas das propriedades de segurança de um grupo descritas anteriormente.
- Colisão – Os membros removidos não poderão obter a nova chave de grupo utilizando o seu conhecimento sobre a chave anterior.

4.1.1.1. GKMP - Group Key Management Protocol

Este protocolo [73] de gestão proporciona a criação e manutenção de uma chave de grupo. Nesta aproximação, o KDC em conjunto com o primeiro membro do grupo cria um *Group Key Packet* (GKP) que contem uma chave de cifra do grupo (GTEK – *Group Traffic Encryption Key*) e a chave de cifra de distrubuição de chaves (GKEK – *Group Key Encryption Key*).

A operação de *join* de um novo membro resume-se à distrbuição do GKP. Quando é necessário refrescar a chave de grupo o KDC gera um novo GKP e cifra-o com o GKEK actual.

Todos os membros do grupo têm conhecimento do GKEK, logo não é possível garantir a propriedade de segurança futura (*forward secrecy*) quando um membro sai do grupo sem obrigar à recriação do grupo, excluindo este membro.

4.1.1.2. LKH - Logical Key Hierarchy

Nesta aproximação [67], o KDC mantem uma árvore de chaves onde os nós da árvore corresponde a chave de cifra de distribuição e as folhas da árvore correspondem a um tuplo <membro, chave de

distribuição>. Cada membro recebe e mantém uma cópia da KEK associada com a sua folha bem como todas as KEKs correspondentes a todos os nós desde o seu nó pai até à raiz da árvore. Esta, por sua vez, contém a chave do grupo.

Numa árvore balanceada, cada membro mantém no máximo ($\log_2$ n) + 1 chaves onde $\log_2$ n é altura da árvore, como se pode ver na próxima figura, onde o membro u_1, tem conhecimento das chaves k_1, k_{12}, k_{14} e k.

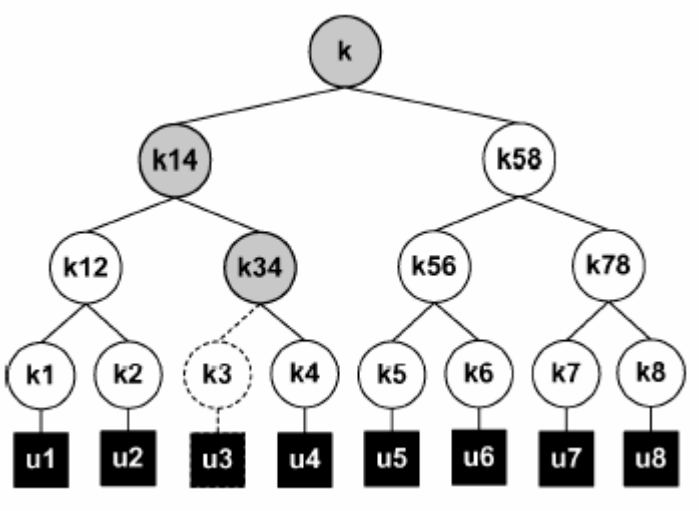

Figura 4-1 Nós afectados quando um novo membro (u_3) entra no grupo.

A operação de join, tomando como exemplo o membro u_3, afecta apenas o caminho desde a raiz até ao seu nó pai, como se pode ver na figura acima.

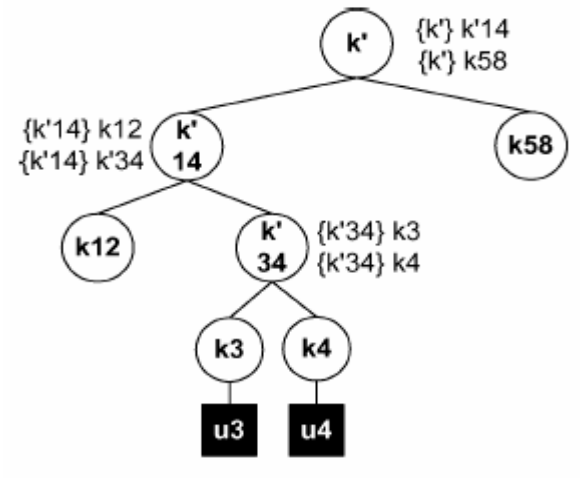

Figura 4-2 Operação de *join* do membro (u_3) no grupo.

Todas as chaves dos nós, desde a raiz até ao seu nó pai, devem ser refrescadas de forma a garantir a propriedade de segurança passada. A mensagem de refrescamento desta operação contém todas as novas KEKs cifradas com as KEKs dos nós filhos numa abordagem *bottom-up*, onde as novas KEKs dos nós intermédios são já cifradas com as KEKs actualizadas dos filhos. Os nós afectados recebem novas KEKs ($k^{'}$, $k^{'}_{14}$,$k^{'}_{34}$) com um custo máximo $2 \cdot (\log_2 n)$ chaves.

Para a remoção de um membro procede-se de forma similar à operação de *join*. As KEKs de todos os nós entre a raiz e o ex-membro precisam de ser refrescadas da mesma forma que na operação join, com a excepção do seu nó pai. Este recebe a sua nova KEK cifrada com a chave dos seus restantes filhos, como se pode ver na figura abaixo.

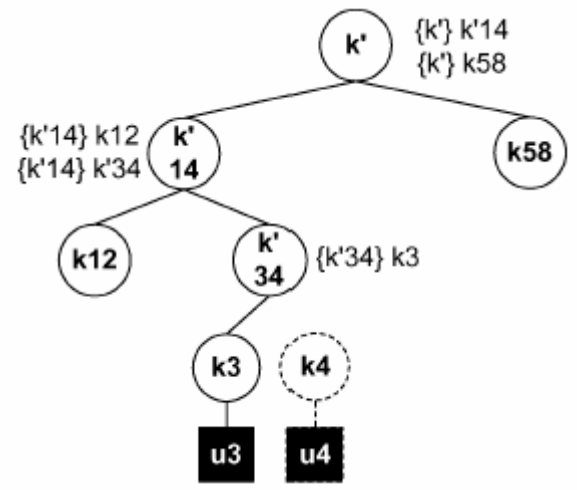

Figura 4-3 Operação de remoção do membro u_4 no grupo.

Este protocolo foi optimizado (proposto por Radia Perlman, durante uma reunião do IETF) com o objectivo de diminuir o envio de mensagens de refrescamento da chave para a operação de *join*. O novo algoritmo, LHK+, utiliza funções *one-way* para gerar as novas chaves. Cada nó utiliza a função para localmente gerar a sua nova chave, evitando o envio de mensagens de refrescamento com as suas novas chaves.

4.1.1.3. OFT – One-way Function Tree

Proposto como uma melhoria ao LHK original [69], este protocolo reduz o tamanho da mensagens de refrescamento da chave para metade, de $2 \cdot (\log_2 n)$ para $(\log_2 n)$. Nesta melhoria um KEK de um nó é gerado localmente em vez de lhe ser atribuido. Esta geração é concretizada através do uso de funções injectivas e funções de consolidação: $k_i = f\,(g(k_{esquerda(i)}), g(k_{direita(i)}))$, onde k_i é o KEK do nó i, f é a função de consolidação e g é a função injectiva.

Tal como no LHK, cada membro guarda $(\log_2 n) + 1$ chaves, mas em vez de guardar as chaves dos nós no caminho entre a raiz e o seu nó pai (*ancestor set*), guarda as chaves do seu irmão e dos irmãos ao longo desse caminho (*sibling set*).

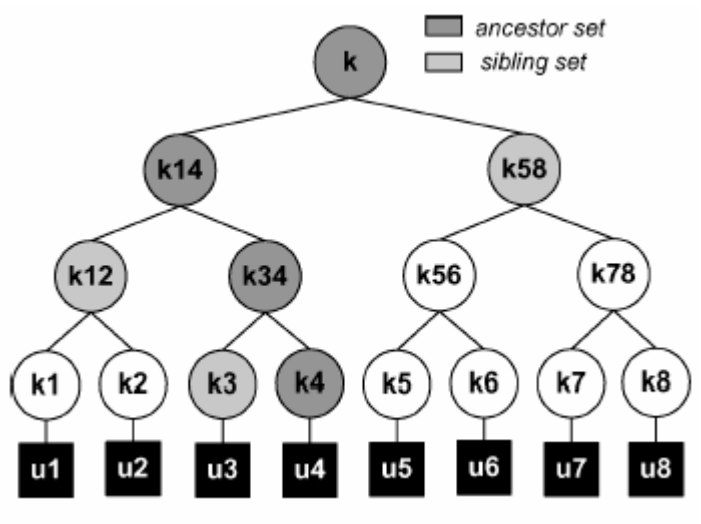

Figura 4-4 Relação dos nós para o membro u_4.

Por exemplo, numa árvore balanceada, o membro u_4 tem conhecimento da sua própria chave k_4 e do resultado da função injectiva para o seu irmão k_3 e para os nós k_{12} e k_{58}. Aplicando a função descrita anteriormente, u_4 gera todas as chaves para os nós k_{34}, k_{14} e k.

A redução do tamanho da mensagem devido a uma alteração ao estado do grupo é conseguida por a mensagem conter só os resultados da função injectiva de cada nó alterado, cifrada com a chave de cada irmão. Um exemplo é o da junção do membro u_3 ao grupo no nó k_{34}. As chaves k_{34}, k_{14} e k terão de ser alteradas e os unicos valores que são transmitidos são o resultado da função injectiva para k_3, k'_{34} e k'_{14} cifrados, respectivamente, com k_4, k_{12} e k_{58}. As novas KEKs pode ser calculadas por todos os membros do grupo: $k'_{34} = f(g(k_3),g(k_4))$, $k'_{14} = f(g(k_{12}),g(k'_{34}))$ e $k' = f(g(k'_{14}),g(k_{58}))$.

4.1.1.4. Hierarchical *a*-ary Tree with Clustering

Este protocolo [70] propõe o agrupamento de vários membros nas folhas da árvore. Ao dividir os *n* membros do grupo em agrupamentos de tamanho *m* e assignar estes agrupamentos a cada folha da árvore, obtemos *n/m* agrupamentos e uma árvore com tamanho ($\log_a (n/m)$).

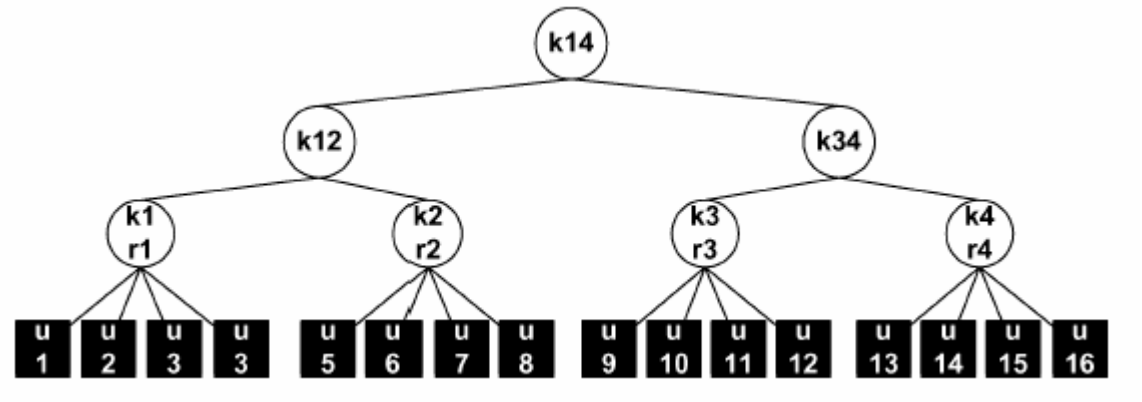

Figura 4-5 Árvore com *a*=2 e com agrupamentos de 4 membros.

Todos os membros do agrupamento têm uma chave privada partilhada com o KDC e partilham a KEK do agrupamento. O KDC utiliza uma semente aleatória (r) como um indexador para uma função injectiva f_r para gerar a chave k_i para o membro i ($k_i = f_r(i)$). Para cada agrupamento apenas é necessário guardar a semente do membro i e a chave do agrupamento. Membros do mesmo agrupamento partilham o conjunto de chaves desde a raiz até à folha onde se encontram, guardando ($\log_a (n/m)$) + 1 KEKs.

A operação de *join* tem um comportamento similar ao dos protocolos anteriores. No entanto, na operação de remoção de um membro, o respectivo agrupamento recebe uma nova chave. Esta nova KEK está cifrada, individualmente, com as KEKs dos restantes membros. Desta forma o KDC efectua m-1 operações de cifra para disseminar a nova KEK do agrupamento aos m-1 membros do respectivo agrupamento. Actualiza ainda as KEKs do caminho entre a raiz e o nó pai do agrupamento, estando as novas chaves cifradas com as KEKs dos nós filhos. Logo a mensagem de refrescamento, quando um membro sai, é m-1 + a ($\log_a$ (n/m)) chaves.

4.1.1.5. CFT – Centralized Flat Tables

Com o objectivo de diminuir o espaço necessário pelo KDC para armazenar as chaves, foi proposta [68] uma extensão ao LHK. O objectivo da proposta é a troca de uma árvore hierárquica por uma tabela. Esta tabela contém uma entrada para a TEK e 2w entradas para as KEKs, onde w é o número de bits do identificador do membro. Existem duas chaves para cada bit no identificador do membro, cada uma associada aos valores possíveis pelo bit.

	TEK	
Bit #0	KEK 0.0	KEK 0.1
Bit #1	KEK 1.0	KEK 1.1
Bit #2	KEK 2.0	KEK 2.1
Bit #3	KEK 3.0	KEK 3.1
	Bit 0	Bit 1

Tabela 4-1 Tabela associada a um membro, quando $w = 4$

Um membro só sabe a chave associada ao estado dos seus bits. Por exemplo, um membro com identificador 1010 sabe $KEK_{0.1}$, $KEK_{1.0}$, $KEK_{2.1}$, $KEK_{3.0}$. No total cada membro sabe w+1 chaves. Quando um membro sai do grupo, todas as chaves conhecidas por este são refrescadas e o KDC envia uma mensagem contendo duas partes. A primeira parte contem a nova TEK cifrada com as KEKs não alteradas, permitindo que todos os membros com uma diferença de pelo menos um bit no identificador consigam obter a nova TEK. Na segunda parte todas as novas KEKs são cifradas com a nova TEK. Desta forma os membros restantes do grupo conseguem refrescar as suas KEKs sem obter conhecimento das KEKs anteriores dos outros membros.

TEK		
$(KEK\ 0.0_{new})\ TEK_{new}$	$(TEK_{new})\ KEK\ 0.1$	ID Bit #0
$(TEK_{new})\ KEK\ 1.0$	$(KEK\ 1.1_{new})\ TEK_{new}$	ID Bit #1
$(KEK\ 2.0_{new})\ TEK_{new}$	$(TEK_{new})\ KEK\ 2.1$	ID Bit #2
$(TEK_{new})\ KEK\ 3.0$	$(KEK\ 3.1_{new})\ TEK_{new}$	ID Bit #3
Bit 0	Bit 1	

Figura 4-6 Mensagem para a remoção do membro 0101.

No entanto este protocolo é vulnerável a ataques de colisão, onde um conjunto de membros excluidos que contêm identificadores com bits complementares podem combinar os seus conjunto de KEKs para obter o novo conjunto de chaves válidas.

4.1.1.6. ELK – Efficient Large-Group Key

O ELK [71] utiliza uma árvore hierarquica e é similar ao OFT, já que a chave do nó pai é gerada a partir das chaves dos seus filhos. Este protocolo utiliza uma função pseudo-aleatória para construir e manipular as chaves utilizadas na árvore.

A função pseudo-aleatória utiliza uma chave K, numa entrada M de tamanho m gerando um resultado de tamanho n e é representada por $PRF_k^{m\to n}(M)$. Aplicando esta função à mesma chave, é possível obter quatro chaves distintas, que são utilizadas em contextos diferentes:

- Para gerar os valores n_1 e n_2 – $k_i^{\alpha} = PRF_{ki}^{n\to n}(1)$
- A chave utilizada para cifrar mensagens de refrescamento – $k_i^{\beta} = PRF_{ki}^{n\to n}(2)$
- Para gerar dicas – $k_i^{\gamma} = PRF_{ki}^{n\to n}(3)$
- Actualização das chaves dos nós – $k_i^{\delta} = PRF_{ki}^{n\to n}(4)$

Este protocolo implementa temporizadores para refrescamento das chaves na àrvore. A chave do grupo é obtida com o uso da função: $k'_G = PRF_{k'G}^{\ n\to n}(0)$ e todas as outras chaves são obtidas usando a chave de grupo como entrada na função: $k'_i = PRF_{k'i}^{\ n\to n}(k_G)$. Como todas as chaves são derivadas, não existem mensagens de *multicast* devido à entrada de novos membros, no entanto podem haver mensagens de *unicast* para membros devido a alterações da sua posição na àrvore como consequência da entrada de novos membros.

A operação de remoção de um membro origina o refrescamento das chaves entre a raiz da àrvore e o seu nó pai. Este refrescamento proporciona a obtenção da nova chave k'_i com $k'_i = PRFC_{LR}^{\ n\to n}(k_i)$, onde $C_{LR} = PRF_{k_{il}}^{\ n\to n1}(k_i) \mid PRF_{k_{ir}}^{\ n\to n2}(k_i)$, k_{il} e k_{ir} são respectivamente os filhos esquerdos e direitos do nó i. O resultado desta operação termina com o envio das mensagens $\{PRF_{k_{il}}^{\ n\to n1}(k_i)\}_{k_{il}^{\beta}}$ e $\{PRF_{k_{ir}}^{\ n\to n2}(k_i)\}_{k_{ir}^{\beta}}$.

O ELK implementa o conceito de indicios. Um indicio é informação de tamanho inferior ao da mensagem de refrescamento da chave, que pode ser utilizado para obter mensagem perdidas de refrescamento de chaves. Toda a chave k_i é gerada a partir de n_1 bits do filho esquerdo e n_2 bits do filho da direita, utilizando um valor para a verificação da chave: $V_{K'} = PRF_{K'}(0)$. O filho da direita consegue (tenta todas as combinações possíveis) obter os n_1 bits do seu irmão e obter a chave $K_i^{'}$. O mesmo se passa para o filho da esquerda, que também consegue saber os n_2 bits do seu irmão e obter a respectiva chave. Usualmente $n_1 < n_2$, fazendo com que o filho da esquerda necessite de mais computações para obter a chave, no entanto devido ao tamanho das contribuições serem diferentes, também necessita de $n_2 - n_1$ bits.

Em resumo, uma mensagem com um indicio é composta pelo valor da verificação da chave ($V_{K'}$) e os $n_2 - n_1$ bits.

4.1.1.7. Comparação entre os protocolos apresentados

A comparação entre os protocolos é sumarizada nos seguintes dois quadros:

Propriedades / Protocolo	Segurança		Resistente a ataques de colisão	Mensagem			Armazenamento	
				Join				
	Passada	Futura		*multicast*	*unicast*	*leave*	KDC	Membro
Simples	S	S	S	*nK*	*K*	*nK*	*nK*	*K*
GKMP	S	N	S	*2K*	*2K*	-	*2K*	*2K*
LKH	S	S	S	*(2d - 1)K*	*(d + 1)K*	*I + 2d K*	*(2n -1)K*	*(d + 1)K*
OFT	S	S	S	*(d + 1)K*	*(d + 1)K*	*I + (d + 1)K*	*(2n - 1)K*	*(d + 1)K*
Clusters	S	S	S	*m – 1 + a $\log_a$ (n/m)*	*$\log_a$ (n/m) + 2*	*m – 1 + a $\log_a$ (n/m)*	*n/m a/(a – 1) + n/m*	*$\log_a$ (n/m) + 2*
FT	S	S	N	*2IK*	*(I + 1)K*	*2IK*	*(2I + 1)K*	*(I + 1)K*
ELK	S	S	S	*0*	*(d + 1)K*	*I + d(n_1 + n_2)*	*(2n – 1)K*	*(d + 1)K*

Tabela 4-2 Comparação dos protocolos centralizados apresentados

Propriedades / Protocolo	Processamento - *Join*		Processamento – *Leave*	
	KDC	Membro	KDC	Membro
Simples	*nE*	*D*	*nE*	*D*
GKMP	*2E*	*2D*	-	-
LHK	*dH + 3dE*	*(d + 1)D*	*2dE*	*dD*
OFT	*(d + 1)H + dX + 3dE*	*(d + 1)D + d(H + X)*	*d(H + X + E)*	*D + d(H + X)*
Cluster	*mPRG + (m + a $\log_a$(n/m))E*	*($\log_a$(n/m)+ 2)D*	*(m – 1)PRG + (m + a$\log_a$(n/m) – 1)E*	*($\log_a$(n/m) + 2)D*
FT	*2IE*	*ID*	*(2I + 1)E*	*(I + 1)D*
ELK	*2(2n – 1)E + 2E + (d + 1)E*	*(d +1)D*	*8dE*	*dD + 5dE*

Tabela 4-3 Comparação dos protocolos apresentados para a computação local

n	Número de membros do grupo
I	Número de bits no identificador do membro
a	
d	Altura da àrvore
m	Tamanho do agrupamento (*cluster*)
H	Função de *hash*
X	Operação *XOR*
E	Operação de cifra
D	Operação de decifra
K	Tamanho da chave em bits

Tabela 4-4 Terminologia utilizada nas tabelas 4-2 e 4-3.

Pode-se concluir que para os protocolos obterem melhores *performances*, estas são conseguidas à custa da não satisfação de todas as propriedades de segurança desejáveis num ambiente de grupo. A melhor relação entre o desempenho e a satisfação dessas propriedades é conseguida pelos protocolos que utilizam o conceito de árvores hierarquicas de chaves.

4.1.2. Arquitectura mista

Esta aproximação divide um grupo de grandes dimensões em sub-grupos com tamanho inferior ao original. Essa divisão implica que cada sub-grupo tem um controlador, minimizando o problema de um ponto de falha e da sobrecarga aplicada ao sistema com um KDC central.

A avaliação dos protocolos nesta classe é efectuada com base nas seguintes considerações:

- Independência da chave relativamente ao passado e futuro
- Controladores de grupo descentralizados
- Refrescamento local das chaves – Alterações ao estado do sub-grupo devem ser tratadas apenas por esse sub-grupo
- Chaves versus Informação – A gestão das chaves deve ser independente da informação recebida/enviada, isto é, o refrescamento das chaves do sub-grupo não deve interferir na comunicação dos seus membros.

- Refrescamento devido a alterações do estado do grupo – Garante as propriedades de segurança passada e segurança futura.
- Tipos de comunicação – Grupos com uma única fonte de informação são chamados *1-para-n* e grupos com multiplas fontes de informação são chamados *n-para-n*.

4.1.2.1. SMKD – Scalable Multicast Key Distribution

Proposto no RFC 1949 [72], este protocolo utiliza um esquema baseado nas àrvores resultantes do protocolo de *routing* de *multicast* Core *Based Tree* (CBT) para disseminar as chaves para o grupo. Devido à autentição necessária pelos *routers* de distribuição da fonte de informação, todos os membros que utilizem esses mesmos *routers* são autenticados e confiáveis para o grupo. Este esquema requer a alteração ao IGMP e assume que é utilizado o CBT para protocolo de *routing*, no entanto não existe solução para garantir a propriedade de segurança futura além da já vista pelo GKMP, a recriação do grupo sempre que sair um membro.

4.1.2.2. Iolus

O Iolus [74] consiste numa solução com hierarquias de agentes que dividem um grupo em vários sub-grupos, geridos por um *Group Security Agent* (GSA). Estes GSAs são, também, agrupados num grupo especial controlado pelo *Group Security Controller*.

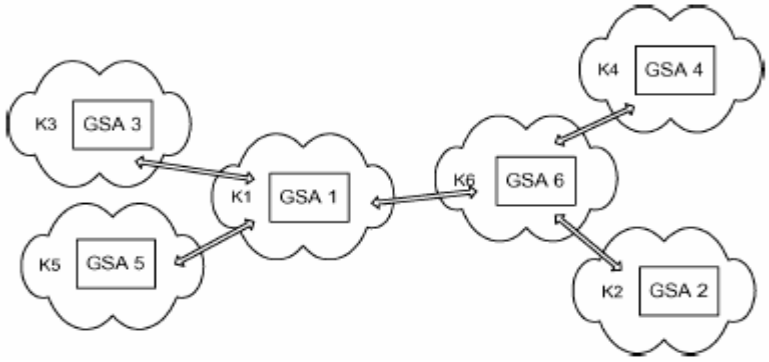

Figura 4-7 Arquitetctura do IOLUS.

Cada subgrupo utiliza uma chave diferente e como tal uma alteração ao estado do sub-grupo é tratado localmente ao sub-grupo.

Apesar do Iolus ser escalável, já que não tem o conceito de uma unidade central de geração e distribuição de chaves, e uma falha no GSA de um sub-grupo abrange só esse sub-grupo, existe a possibilidade de congestionamento no GSA devido à necessidade deste ter que gerir as chaves do sub-grupo e traduzir a informação que atravesse os vários sub-grupos com o qual está ligado.

4.1.2.3. DEP - Dual-Encryption Protocol

Com o objectivo de resolver a problemática da confiança em entidades terceiras, este protocolo [75] sugere uma hierarquia por sub-grupos, dentro do qual existe um controlador denominado *sub-group manager* (SGM).

Existem quatros tipos diferentes de chaves, três KEKs e uma *Data Encryption Key* (DEK):

- A chave KEK_{i1} é partilhada entre o SGM_i e os membros do seu sub-grupo
- A chave KEK_{i2} é partilhada entre o controlador global do grupo e os membros do sub-grupo *i* com a exclusão do SGM_i
- A chave KEK_{i3} é partilhada entre o controlador de grupo e o SGM_i

Para a disseminação de chaves do controlador do grupo para os membros de cada sub-grupo, este envia uma mensagem para o SGM com o resultado da cifra da chave de comunicação (DEK) com a chave KEK_{i2}, cifrado com a chave KEK_{i3}. Ao receber esta mensagem, o SGM decifra a mensagem, obtendo {DEK} KEK_{i2} e cifra este resultado com a chave KEK_{i1}, que por sua vez envia-a para os membros do sub-grupo. Por fim, os membros do sub-grupo, decifram a mensagem utilizando, respectivamente, as chaves KEK_{i1} e KEK_{i2} para obter a DEK. A chave do grupo não pode ser obtida sem conhecimento destas duas chaves.

Quando existe uma alteração ao estado do sub-grupo *i* o SGM_i altera a chave KEK_{i1} e envia-a aos restantes membros do sub-grupo. No entanto, um membro que tenha saido do grupo ainda pode decifrar a comunicação do grupo enquando não for gerada uma nova DEK, um comportamento que compromete a propriedade de segurança futura.

4.1.2.4. MARKS

A solução proposta em [76] sugere a divisão temporal da informação a ser cifrada, em que cada uma dessas divisões é cifrada com chaves diferentes. As chaves de cifra são as folhas de uma àrvore binária, a raiz e nós intermédios são sementes.

Utilizando uma função *f*, do tipo *MD5*, a criação da àrvore efectua-se da seguinte maneira:

1. A altura da àrvore *D* é escolhida – a altura influencia o número total de chaves *N* ($N = 2^D$).
2. A semente da raiz, $S_{0,0}$, é escolhida de forma aleatória. Na notação $S_{i,j}$, *i* representa a altura da àrvore e *j* representa o número da chave na altura *i*.
3. Dois nós intermédios (o nó esquerdo e nó direito) são criados. O nó esquerdo é criado através da aplicação da função *f* à transposição para a esquerda de um bit de $S_{0,0}$ ($S_{1,0} = f(\text{te}(S_{0,0}))$). O

nó direito é criado através da aplicação de *f* à transposição para a direita de um bit de $S_{0,0}$ ($S_{1,1}$ = *f* (td($S_{0,0}$))).

4. Repetir o passo 3 até atingir a altura da àrvore desejada.

Os utilizadores que desejem obter acesso ao grupo obtêm as sementes necessárias para gerar as chaves necessárias. Como se pode ver na próxima figura, um utilizador que deseje participar no grupo durante o periodo compreendido entre a terceira e setima divisão temporal, iria necessitar de apenas duas sementes: $S_{3,3}$ e $S_{1,1}$ de forma a gerar, respectivamente, K_3 e K_4 a K_7.

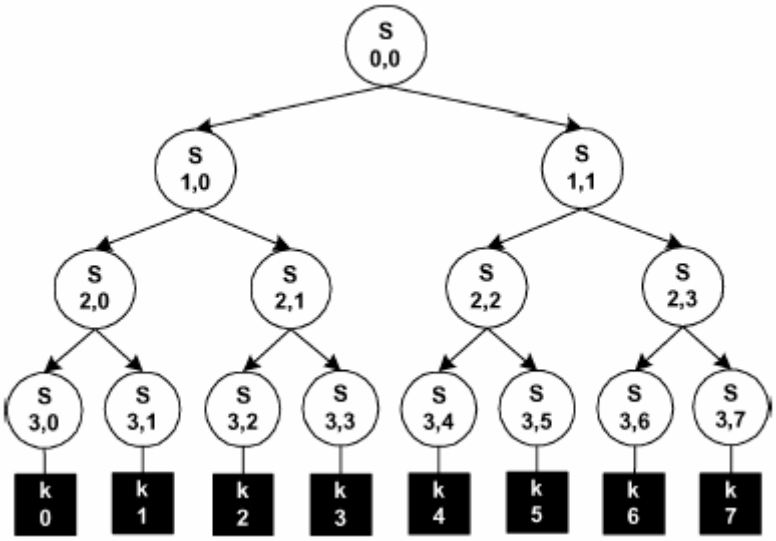

Figura 4-8 Árvore binária do protocolo MARKS.

4.1.2.5. CS - Cipher Sequences

Apresentado [77] como um *framework* para segurança em redes *multicast*, é baseado no uso de sequências de cifras. Uma função *f*(*S*, *a*) é uma sequência de cifras se tiver três caracteristicas:

- Existe uma sequência de *n* elementos, tal que $a_{1\leq i\leq n}$
- Existe uma sequência de $n + 1$ elementos, como por exemplo, $S_{0\leq i\leq n}$ onde S_0 é o valor inicial e $S_i = f(S_{i-1}, a_i)$ para todo o $i > 0$.
- Para todo tuplo (i, j) onde $i < j$, existe uma função $h_{i,j}$ tal que $S_i = h_{i,j}(j)$.

Este protocolo é orientado para um esquema *one-to-many* e como tal o grupo de *multicast* é colocado numa àrvore em que a raiz é a fonte de informação e as folhas são os receptores.

O processo de propagação de uma mensagem é efectuado com a seguinte lógica:

- S_0 é a informação a ser transmitida e todo o nó N_i tem um valor a_i associado.
- Todo o N_i, ao receber o valor S_j do seu pai, computa $S_i = f(S_j, a_i)$ e envia S_i aos seus filhos
- Para as folhas foi assignado a função $h_{0,n}$, que permite obter S_0 de S_n já que $S_0 = h_{0,n}(S_n)$

A proxíma figura ilustra um exemplo, em que a raiz calcula $S_1 = f(S_0, a_1)$ e envia S_1 a N_1 , o nó N_1 calcula $S_2 = f(S_1, a_6)$ e envia S_2 para N_2. Este nó calcula $S_3 = f(S_2, a_7)$ e envia S_3 para a folha L_3 e, finalmente, a folha L_3 obtem $S_0 = h_{0,3}(S_3)$.

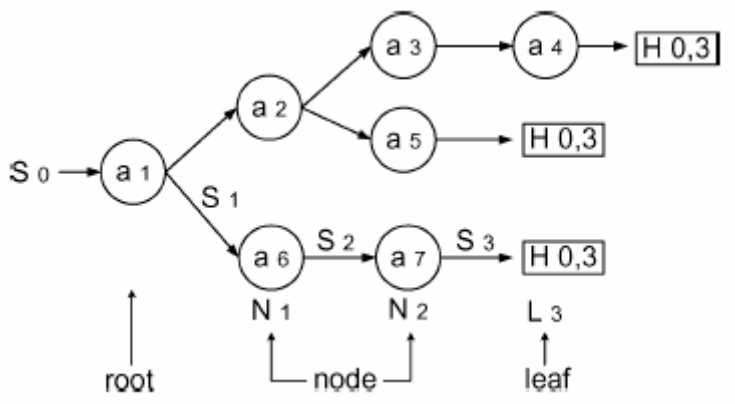

Figura 4-9 Exemplo da execução do protocolo CS.

Uma folha pode ser constituida por vários membros e todos os membros partilham a função $h_{0,n}$. Quando existir uma alteração ao estado de uma folha, o seu nó pai recebe um novo valor a_n' e todos os membros da folha recebem a nova função $h_{0,n}'$. No caso da remoção de um membro, a nova função não é enviada para esse membro, garantindo a propriedade de segurança futura.

4.1.2.6. IGKMP – Intra-Domain Group Key Management Protocol

Apresentado [78] com o objectivo de dividir àreas de forma administrativa, este protocolo consiste num distribuidor de chaves do domínio (*Domain Key Distributor – DKD*) e em vários distribuidores de chaves de àrea (*Area Key Distributor – AKD*) onde cada AKD é responsável por uma àrea.

A chave do grupo é gerada pelo DKD e distribuida para os membros através dos AKDs. Os gestores da chave (DKD e AKDs) partilham um grupo *multicast* para comunicação (all-KD-group), logo não existe tradução de chaves entre àreas.

Contudo, devido à existência de um servidor central, que gera a chave e gera os AKDs, todo o grupo pode ficar comprometido caso o DKD falhe, tal como, caso falhe um AKD, todos os membros pertencentes a àrea gerida por este ficam impossibilitados de comunicar com o resto dos grupos.

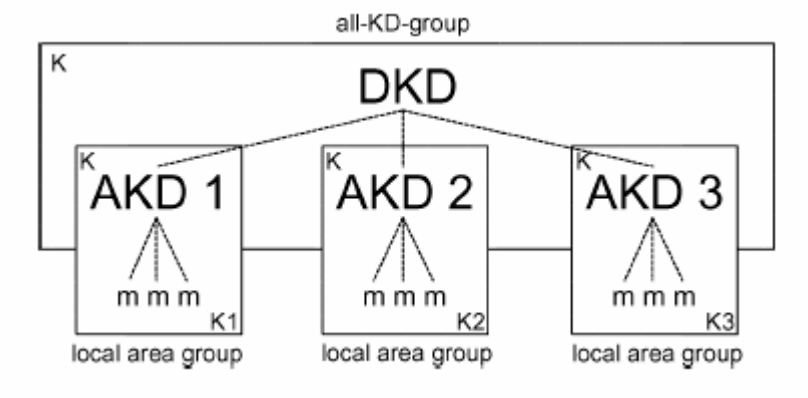

Figura 4-10 Arquitectura do IGKMP

4.1.2.7. Kronos

Esta aproximação [79] assenta em temporizadores para actualizar a chave do grupo. Qualquer alteração ao estado do grupo não é relevante para o refrescamento da chave.

Apesar do Kronos poder ser utilizado no conceito de àreas administrativas, este comporta-se de forma diferente do IGKMP, já que o DKD não cria a chave de grupo. Cada AKD gera, de forma independente, uma chave para a sua àrea e transmite-a aos seus membros.

O funcionamento deste protocolo baseia-se na necessidade dos AKDs gerarem novas chaves quando expira um temporizador – sendo necessária a sincronização dos relógios dos AKDs utilizando, por exemplo, o NTP [80] – e de os AKDs concordarem com o DKD em dois parametros, K e R_0, onde R_0 é o valor inicial e K é a chave inicial. Para gerar a chave de grupo R_1, os AKDs aplicam a função de cifra E a R_0 utilizando K como chave original ($R_1 = E_k(R_0)$). O mesmo algoritmo se aplica para as proximas gerações de chaves: $R_{i+1} = E_k(R_i)$ com $i > 0$.

Apesar do Kronos não apresentar um servidor central de geração de chaves, esta geração é baseada em chaves anteriores, comprometendo a segurança do grupo caso seja descoberta uma chave anterior.

4.1.2.8. Hydra

O Hydra [81] consiste num grupo dividido em sub-grupos, e um servidor designado por servidor Hydra (HS) que controla cada sub-grupo.

A alteração ao estado de um sub-grupo HS_i tem como consequencia a geração de uma chave, e o envio dessa nova chave para todos os HS_j que estejam involvidos no grupo. Caso um HS esteja inacessível, o resto de grupo prossegue a comunicação sem perdas de *performance*.

A garantia da sincronia e distribuição da nova chave para todos os HSs é efectuada pelo protocolo SGKDP – *Synchronized Group Key Distribution Protocol*. Este protocolo garante a existência de apenas um HS válido para a geração da chave do grupo.

4.1.2.9. Comparação dos protocolos mistos

Tal como foi apresentado anteriormente, a comparação deste protocolos é resumida na seguinte tabela:

Propriedades / Protocolo	Indepêndencia da chave	Descentralizado		Refrescamento local	Informação vs. Chaves	Refrescamento das chaves	Tolerênacia a falhas	Tipo de comunicação
		Gestão	KDC					
SMKD	S	S	S	N	S	N	N	*n*-para-*n*
Iolus	S	S	S	S	N	S	S	*1*-para-*n*
DEP	S	N	S	N	S	N	N	*n*-para-*n*
CS	S	N	N	N	S	S	N	*1*-para-*n*
Marks	N	S	-	N	S	N	S	*n*-para-*n*
Kronos	N	S	S	N	S	N	S	*n*-para-*n*
IGMKP	S	S	N	N	S	S	N	*n*-para-*n*
Hydra	S	S	S	N	S	S	S	*n*-para-*n*

Tabela 4-5 Comparação dos protocolos mistos apresentados

4.1.3. Arquitectura Descentralizada

A característica dos protocolos que suportam esta arquitectura consiste em não terem um controlador de grupo, onde a chave de grupo é gerada de forma contributória ou apenas por um membro. Quando a geração está a cargo de um único membro, apesar da aproximação ser tolerante a falhas, é preciso notar quais as capacidades de processamento e utilização de números aleatórios. A geração de chaves de forma contributória tem um acréscimo na carga de gestão, já que para garantir a robustez dos protocolos é necessário que todos os membros do grupo tenham conhecimento das alterações efectuadas ao estado do grupo. Nesta última forma de geração de chaves também é relevante relacionar o aumento linear do número de membros com o aumento do tempo de processamento e requisitos de comunicação.

A comparação destes protocolos é baseada nos seguintes parametros:

- Número de rondas – O número de rondas do protocolo deve ser o minimo de forma a minimizar os custos de processamento e comunicação.
- Número de mensagens
- Processamento durante a inicialização
- Chaves *Diffie-Hellman* (DH)

4.1.3.1. BD – Burmester and Desmedt Protocol

Proposto em [80], é muito eficiente em termos computacionais e é executado em três rondas:

1. O membro m_i gera o seu expoente secreto r_i e faz o *broadcast* de $= \alpha^{r_i}$
2. O membro m_i calcula e envia em *broadcast* $X_i = (Z_{i+1}/Z_{i-1})^{r_i}$
3. A chave de grupo é calculada através de: $k = Z_{i-1}^{nr_i} \cdot X_i^{n-i} \cdot X_{i+1}^{n-2} \cdots X_{i-2} \bmod p$

Este protocolo requer $n + 1$ exponenciações por membro, com o custo do envio de $2n$ mensagens de *broadcast*.

4.1.3.2. GDH (Group Diffie-Hellman)

Tal como o nome indica este modelo é uma extensão ao modelo Diffie-Hellman, e consiste na contribuição de todos os membros para a geração da chave simétrica de grupo.

Para tal a geração da chave K é feita através de uma função $f(N_1,...,N_n)$ onde $f()$ é uma função de hash e N_i é a contribuição de cada membro. O método de geração da chave deve garantir pelo menos os seguintes pontos:

- Qualquer membro que contribua com N_i pode calcular K.
- Não é possível obter qualquer tipo de informação sobre K sem ter conhecimento de pelo menos um N_i.
- Todos os N_i são secretos, isto é se o membro i é honesto então nem com a junção dos outros membros se pode obter alguma informação sobre N_i.

Este modelo é facilmente estendido, tal como demonstrado em [22], de modo a serem obtidas as seguintes propriedades:

- Autenticação da chave
- *Perfect Forward Secrecy* (PFS) [21]
- Resistência a ataques de chaves conhecidas [21]

- Confirmação e integridade da chave

Devido aos objectivos do GDH, a sua integração com as operações do SCG é essencial. Vamos demonstrar o mapeamento das várias operações de um SCG no GDH.

Na criação do grupo efectua-se o início da geração e distribuição da chave de grupo (*Initial Key Agreement* - IKA). Este início está dividido em duas fases. Na primeira fase reúnem-se as contribuições de todos os elementos, uma por ronda. Na ronda i ($i \in [1,n-1]$), M_i envia a M_{i+1} um número de i de valores. Destes, $i-1$ são valores intermédios e o valor restante é o cardinal. O valor cardinal CRD_i é simplesmente o gerador elevado a todos os expoentes secretos gerados:

$$CRD_i := \alpha^{\Pi(Np|p\in[1,i])}$$

Seja $INT_{i,j}$ o valor intermédio j na ronda i ou seja, CRD_i sem o expoente j:

$$INT_{i,j} := \alpha^{\Pi(Np|p\in[1,i]\wedge p\neq j)} \quad ,j \in [1,i]$$

Na ronda i, M_i faz os seguintes cálculos:

1. Gera o expoente privado N_i
2. $INT_{i,j} = (INT_{i-1,j})^{Ni}$ para todo o $j \in [1,i-1]$
3. $INTi,i = CRD_{i-1}$
4. $CRDi = (CRD_{i-1})^{Ni}$

No total M_i gera i valores intermédios (cada um com (i - 1) expoentes) e um valor cardinal contendo i expoentes. Por exemplo, M4 recebe o conjunto:

$$\{\alpha^{N_1 N_2 N_3}, \alpha^{N_1 N_2}, \alpha^{N_1 N_3}, \alpha^{N_3 N_2}\}$$

E envia o seguinte conjunto:

$$\{\alpha^{N_1 N_2 N_3 N_4}, \alpha^{N_1 N_2 N_3}, \alpha^{N_1 N_2 N_4}, \alpha^{N_1 N_3 N_4}, \alpha^{N_3 N_2 N_4}\}$$

Na segunda fase, na ronda (n – 1) quando o conjunto é recebido por M_n , o valor cardinal é $\alpha^{N_1 \ldots N_{n-1}}$. Logo, M_n é o primeiro membro do grupo a calcular a chave do grupo K_n. Como resultado M_n calcula os valores intermédios restantes e faz um *broadcast* desse conjunto para todos os membros do grupo.

Em conclusão, tem-se os seguintes custos:

Tipo	Custo
Nº de rondas	n
Nº de mensagens	n
Tamanho combinado das mensagens	$(n-1)(n/2+2)-1$
Cálculos por M_i	$(i+1)$ para $i < n$, n para M_n
Total de cálculos	$((n+3)n)/2-1$

Tabela 4-6 Custos do IKA do GDH

M_i M_{i+1}

$\{ \alpha^{\Pi(Np \mid p \in [1,i] \wedge p \neq j)} \mid j \in [1,i] \}$, $\alpha^{\Pi(Np \mid p \in [1,i])}$ →

Primeira fase (Unicast): ronda i; $i \in [1,n-1]$

M_* M_n

$\{ \alpha^{\Pi(Np \mid p \in [1,n] \wedge p \neq j)} \mid j \in [1,n] \}$ ←

Segunda fase (Broadcast): ronda n

Figura 4-11 Operação inicial de geração de chave de grupo

Após esta operação inicial, é eleito um controlador de grupo. O controlador de grupo pode ser qualquer membro desde que tenha conhecimento do último *broadcast* com o conjunto dos valores intermédios. A flexibilidade originada pelo facto de o controlador de grupo poder ser qualquer membro proporciona alta disponibilidade ao protocolo. Tipicamente o controlador de grupo é o último membro do grupo que foi adicionado.

As restantes operações que iremos ver em seguida são chamadas *Auxiliar Key Agreement* – AKA.

Na operação de adição de um novo membro, o protocolo GDH procede da seguinte forma:

1. Estende a primeira fase do IKA, tendo o controlador de grupo de gerar um novo expoente N_c' e criar uma nova mensagem que envia ao novo membro. N_c'N_c é utilizado no lugar de N_c para prevenir que novos membros e terceiros possam descobrir a antiga chave do grupo. É obvio que na construção da mensagem o controlador de grupo altera o seu antigo valor N_c por N_c'N_c nos expoentes dos valores intermédios.
2. O novo membro continua o processo, prosseguindo a segunda fase do IKA.

M_c M_{n+1}

$\{ \alpha^{Nc'\Pi(Np|p\in[1,n]\wedge p\neq j)} | j \in [1,n]\}$, $\alpha^{Nc'\Pi(Np|p\in[1,n])}$ →

Primeira fase (Unicast): ronda 1; $N_c \leftarrow N_c'N_c$

M_* M_{n+1}

$\{ \alpha^{Nc'\Pi(Np|p\in[1,n+1]\wedge p\neq j)} | j \in [1,n+1]\}$ ←

Segunda fase (Broadcast): ronda 2

Figura 4-12 Operação de *Join*

A operação de *Merge* é um caso especial da adição de *m* novos membros. É possível construir esta operação à custa da operação de adição de um membro mas traz as seguintes desvantagens:

- Tem um custo de 2*m* rondas, já contando com o custo de m rondas de *broadcast*
- *m* exponenciais para cada novo membro do grupo

A alternativa é encadear a adição dos *m* novos membros numa sequência de mensagens que percorre todos os novos membros numa certa ordem. Esta alternativa obtém uma única mensagem de *broadcast* e permite adiar esta mensagem até ao último membro. Com esta nova versão tem-se apenas *m* – 1 rondas.

M_{n+i} M_{n+i+1}

$\{ \alpha^{Nc'\Pi(Np|p\in[1,n+i]\wedge p\neq j)} | j \in [1,n+i]\}$, $\alpha^{Nc'\Pi(Np|p\in[1,n+i])}$ →

Primeira fase (Unicast): ronda i ($0\leq i\leq m$) (caso (i=0) { $M_{n+i} := M_c$; $N_c \leftarrow N_c'N_c$ })

M_* M_{n+m}

$\{ \alpha^{Nc'\Pi(Np|p\in[1,n+m]\wedge p\neq j)} | j \in [1,n+m]\}$ ←

Segunda fase (Broadcast): ronda m + 1

Figura 4-13 Operação de Merge

A operação de *Leave* é algo similar à operação de *Join,* e consiste na execução da última ronda do IKA. Tal como no *Join* é gerado um novo expoente N_c' e construída uma nova mensagem de *broadcast* utilizando $N_c'N_c$ ao invés de N_c.

Consideremos M_d, o membro a ser excluído e que $d \neq c$. Devido à nova sub chave ser $\alpha^{Nc'\Pi(Np|p\in[1,n]\wedge p\neq d)}$ e o membro d estar excluído, este membro é incapaz de calcular a nova chave $K_{nova} = \alpha^{Nc'\Pi(Np|p\in[1,n])}$. É de notar dois aspectos, o primeiro consiste na utilização de N_d para a geração da nova chave, o outro consiste no facto de que se for o membro M_c a ser excluído do grupo, então qualquer outro membro que tenha a ultima mensagem de *broadcast* pode ocupar o lugar de controlador de grupo.

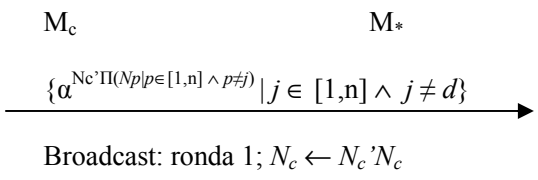

Figura 4-14 Operação de Leave

A operação de *Partition* é praticamente idêntica à operação de *Leave,* em que a única diferença consiste no menor número de cálculos e o facto da mensagem de *broadcast* ter menos sub chaves. Caso o resultado da operação *Partition* seja um sub grupo então a solução passa por eleger um controlador de grupo e depois proceder da forma já descrita.

Uma outra operação que este protocolo oferece mas que não tem mapeamento para as operações de um SCG é a operação de refrescamento da chave de grupo. Existem duas razões para a implementação desta operação:

- Limita o tempo de vulnerabilidade em caso de perca, extravio ou descoberta da chave de grupo.
- Limita a quantidade de cifra de texto disponível para tentar descobrir a chave do grupo.

Fazer o refrescamento da chave de grupo não é suficiente, também é necessário fazer o refrescamento das contribuições de cada elemento do grupo. Chega-se assim ao seguinte protocolo: o membro M_h, que corresponde ao membro que fez o refrescamento há mais tempo, gera uma nova contribuição e volta a executar a ronda de *broadcast*.

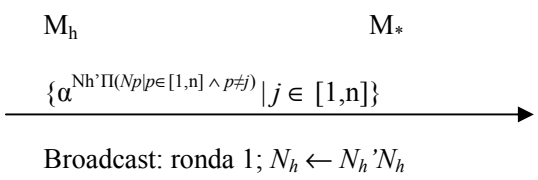

Figura 4-15 Operação de Partition

Existem extensões a este protocolo de forma a reduzir alguns custos. Por exemplo, temos o TGDH [23] que introduz o conceito de arvore ao protocolo e reduz o número de cálculos efectuados por membro de

O(n) para O(log n) e outro protocolo, descrito em [24], que reduz o numero de mensagens trocadas entre os membros.

Em seguida analisa-se uma extensão a este protocolo, o A-GDH [xx] de forma a suportar autenticidade da distribuição da chave de grupo.

4.1.3.3. A-GDH

O A-GDH é uma extensão ao GDH original que foi desenvolvido, tal como o GDH, pelo projecto CLIQUES [32].

Já foi referido que um dos problemas de segurança do DH original era o *Man in the Middle* e tal como no protocolo base, o GDH também está sujeito ao mesmo problema. A solução para o problema original consiste em autenticar as trocas das mensagens que contém os expoentes. Enquanto para o problema inicial a assinatura das mensagens de troca de expoentes não tinha grande impacto na sua *performance*, para a sua extensão em grupo, o mesmo já não é verdade. Para lidar com a questão da autenticidade e integridade da chave de grupo surgiu o A-GDH (*Authenticated* GDH) e suas variantes.

Este protocolo consiste numa extensão ao GDH e além dos requisitos enumerados no GDH necessita que o membro M_n partilhe com cada M_i uma chave diferente K_{in}. Um exemplo para a geração desta chave consiste em:

1. x_i – expoente secreto eleito por cada membro i com $1 < x_i < q$-1
2. $\alpha^{xi} \bmod p$ – chave publica do membro i
3. K_{in} – chave partilhada que corresponde a $F(\alpha^{xi.xn} (\bmod p))$ com $i \in [1, n\text{-}1]$

A extensão ao IKA incial do GDH é esquematizada da seguinte forma:

1. Na ronda 1, $n - 1$, procede-se de forma igual ao GDH

2. Na ronda n, M_n escolhe um expoente Nn, efectua a operação de multiplicação enunciada no GDH e também efectua a operação de multiplicação do expoente em que adiciona a cada valor a chave partilha com o membro i.

M_i M_{i+1}

$\{ \alpha^{\Pi(Np \mid p \in [1,i] \wedge p \neq j)} \mid j \in [1,i] \}$, $\alpha^{\Pi(Np \mid p \in [1,i])}$ →

Primeira fase (Unicast): ronda i; $i \in [1,n-1]$ (operação igual ao GDH original)

M_* M_n

← $\{ \alpha^{\Pi(Np \mid p \in [1,n] \wedge p \neq j) * Kjn} \mid j \in [1,n] \}$

Segunda fase (Broadcast): ronda n

Figura 4-16 Operação de IKA do A-GDH

Após a recepção, cada M_i calcula: $\{ \alpha^{\Pi(Np \mid p \in [1,n] \wedge p \neq j) * Kjn} \mid j \in [1,n] \} * \alpha^{1/Kin * Ni} = \alpha^{N1 \ldots Nn} = S_n$ (chave de sessão partilhada pelo grupo).

Demonstramos a seguir um exemplo do protocolo para 4 participantes:

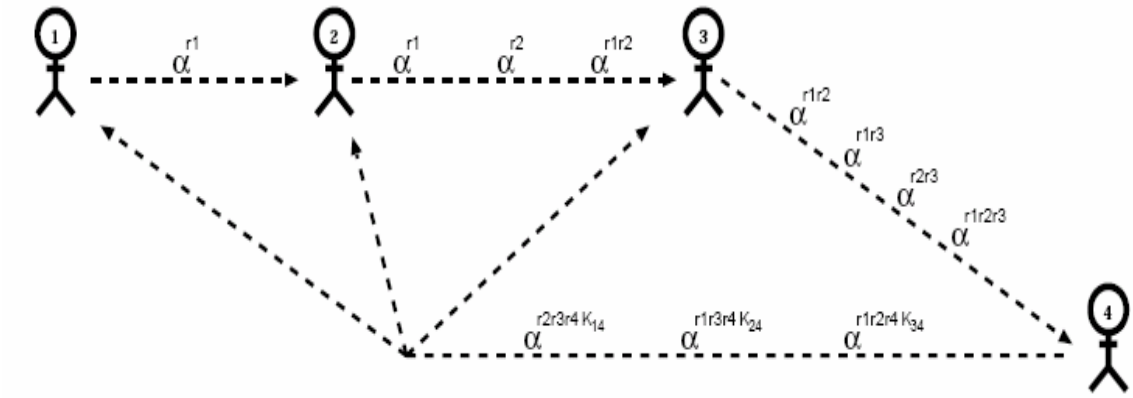

Figura 4-17 IKA do protoclo A-GDH.

As operações IKA enunciadas no GDH sofrem a mesma alteração que sofreu o AKA, logo a extensão ao GDH inicial é directa e simples.

Com estas alterações é possível obter-se um sistema de geração e distribuição de chaves contributivas autenticadas.

Uma outra extensão efectuada ao GDH foi o SA-GDH [46] que consiste na autenticação completa da chave de grupo. A maior alteração à *priori* ao A-GDH consiste em que cada M_i necessita de duas chaves partilhadas com todos os outros membros.

Na próxima figura nota-se as diferenças no desenrolar desta extensão:

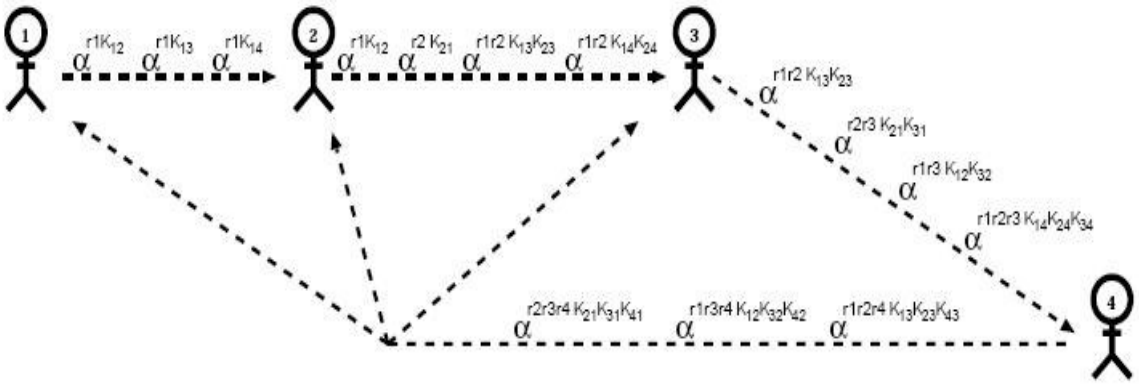

Figura 4-18 IKA do protocolo SA-GDH.

Com esta alteração o controlador de grupo e todos os membros do grupo tem a certeza da geração dos expoentes por parte de todos os membros.

Estas extensões ao protocolo inicial foram quebradas em [39], e em [40] por Olivier Pereira que enuncia um novo protocolo que elimina os problemas descobertos pelo mesmo, no entanto o protocolo anunciado não permite entradas e saídas dinâmicas do grupo.

4.1.3.4. Octopus

Tal como no GDH, este protocolo baseia-se em DH para a criação da chave. O Octopus [82] divide um grupo de *n* elementos em quatro sub-grupos com *n*/4 elementos cada. Os subgrupos geram um valor DH intermédio ($Isubgrupo = \alpha^{u_1 \cdots u_{n/4}}$, onde u_i é a contribuição do membro *i*) e trocam entre si esse valor. Cabe ao lider de cada subgrupo receber as contribuições dos membros e trocar os valores intermédios com os lideres dos outros subgrupos.

A operação de criação da chave é constituída pelos seguintes passos:

1. Os lideres de cada subgrupo (*A, B, C, D*) obtêm as contribuições dos membros dos respectivos subgrupos
2. Os lideres criam os valores DH intermédios I_a, I_b, I_c, I_d e trocam esses valores entre si, o lider *A* troca o seu valor com o lider *B*, o lider *C* troca o seu valor com o lider *D*, resultando $I_a \cdot I_b$ *e* $I_c \cdot I_d$. Finalmente o lider *A* troca o resultado $I_a \cdot I_b$ com o resultado $I_c \cdot I_d$ do lider *C* e o lider *B* troca o resultado $I_a \cdot I_b$ com o resultado $I_c \cdot I_d$ do lider *D* obtendo a chave final $I_a \cdot I_b \cdot I_c \cdot I_d$.
3. Cada lider envia para cada membro do seu subgrupo $\alpha^{(Ia \cdot Ib \cdot Ic \cdot Id)/u_i}$, onde $i = 1 \cdots (n - 4)/4$.
4. Cada membro calcula localmente a chave do grupo

4.1.3.5. CKA - Conference Key Agreement

Proposto em [83], este protocolo consiste numa função $K = f(N_1, h(N_2),\cdots, h(N_n))$, onde f é uma função de combinação (do tipo da função MAC), h é uma função injectiva, n é o número de membros do grupo e N_i é a contribuição do membro i.

O protocolo desenrola-se com o *broadcast* em canal aberto das contribuições de $n - 1$ membros. O líder do grupo, por exemplo U_1, cifra a sua contribuição N_1 com as chaves publicas dos $n - 1$ membros. Todos os membros do grupo obtêm essa contribuição, decifram-na e obtêm a chave final do grupo.

4.1.3.6. DLKH – Distributed Logical Key Hierarchy

Este protocolo [84] é uma aproximação distribuida ao LKH apresentado anteriormente, onde é abolido o controlador de grupo e a hierarquia da àrvore é gerada pelos seus membros, fazendo com que nenhum membro saiba todas as chaves.

O uso da noção de as sub-àrvores acordarem uma chave é a base deste protocolo. Sejam duas sub-àrvores L e R, cujas chaves são k_L *e* k_R e dois lideres, m_l e m_r respectivamente:

1. O membro m_l escolhe uma nova chave k_{LR}, e envia-a para m_r através de um canal seguro.
2. O membro m_l cifra a nova chave k_{LR} com a sua chave k_L e envia-a para os seus filhos. Por sua vez o membro m_r efectua o mesmo procedimento com a sua chave k_R.
3. Todos os membros das sub-àrvores recebem a mesma chave.

Na próxima figura podemos ver um exemplo da execução deste protocolo:

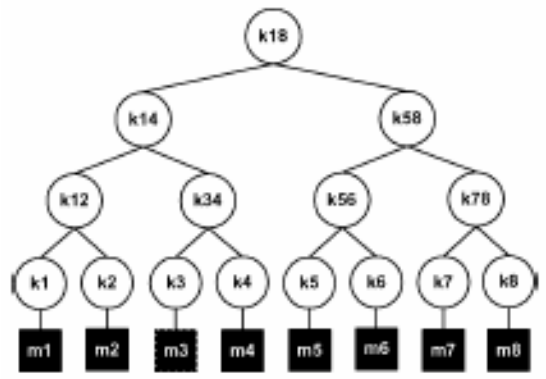

Figura 4-19 Hierarquia do DLKH.

1. Os membros m_1 e m_2 concordam com a chave k_{12}, os membros m_3 e m_4 concordam com a chave k_{34}, os membros m_5 e m_6 concordam com a chave k_{56}, os membros m_7 e m_8 concordam com a chave k_{78}.
2. Os membros m_1, m_2, m_3 e m_4 concordam com a chave k_{14}, os membros m_5, m_6, m_7 e m_8 concordam com a chave k_{58}.
3. Os membros m_1, m_2, m_3, m_4, m_5, m_6, m_7 e m_8 concordam com a chave k_{18}.

Neste caso o protocolo demora três rondas e cada membro tem conhecimento de três chaves. Na generalidade este protocolo demora $\log_2 n$ rondas e cada membro tem conhecimento de $\log_2 n$ chaves.

4.1.3.7. DOFT – Distributed One-way Function Tree

O DOFT baseia-se no uso das OFT vistas anteriormente, distribuidas numa árvore hierarquica lógica. A diferença entre esta versão [85] e o OFT é o facto de não haver uma entidade central controladora e cada membro do grupo ser responsável pelo controlo de acessos e geração da chave de grupo.

Os membros geram a sua própria chave e enviam o resultado da função injectiva para os seus filhos. Tal como na árvore centralizada, todos os membros sabem as chaves do caminho desde a raiz da árvore até si e sabem, também, os resultados da função injectiva desde a raiz da árvore até aos seus filhos directos.

4.1.3.8. DHLKH – Diffie-Hellman Logical Key Hierarchy

Com o objectivo de minimizar o número de chaves conhecidos pelo controladores do grupo, este protocolo [86] diferencia-se do anterior pelo uso de Diffie-Hellman na árvore. A chave de um nó é gerada através dos seus filhos: $k = \alpha^{k_l k_r} \bmod p$.

Usando como exemplo a figura 4-19, obtemos as chaves $k_{12} = \alpha^{k_1 k_2} \bmod p$, $k_{34} = \alpha^{k_3 k_4} \bmod p$, $k_{56} = \alpha^{k_5 k_6} \bmod p$, $k_{78} = \alpha^{k_7 k_8} \bmod p$ e seguindo o algoritmo temos, também, $k_{14} = \alpha^{k_{12} k_{34}} \bmod p$, $k_{58} = \alpha^{k_{56} k_{78}} \bmod p$, $k_{18} = \alpha^{k_{14} k_{58}} \bmod p$.

4.1.3.9. DFT – Distributed Flat Table

Extendido pelo criador original das *flat tables*, esta proposta [68] elimina o uso do controlador de grupo. Neste protocolo, nenhum membro conhece todas as chaves em simultâneo, apenas as necessárias para poder enviar e receber mensagens cifradas. No entanto, esta proposta tem o problema da sincronização das chaves quando um membro se junta a um grupo, já que este tem que contactar um certo número de membros para obter todas as chaves de que necessita.

4.1.3.10. Comparação dos protocolos descentralizados

Apresenta-se de seguida a tabela que resume a comparação dos protocolos descritos anteriormente:

Propriedades / Protocolo	Nº rondas	Nº menssagens		Chave DH	Inicialização	
		Multicast	Unicast		Líder	Outros
BD	3	$2n$	0	N	-	$(n + 1)Ex$
GDH	n	n	n - 1	S	-	$(i + 1)Ex$
Octupus	$2(n – 4)/4 + 2$	0	$3n – 4$	S	$(2(n – 4)/4 + 2)Ex$	$4Ex$
CKA	3	n	n - 1	N	$H + (n – 1)(E + H) + M$	$D + nH + M$
DLKH	3	1	n	N	$\log_2 nEx$	$\log_2 nD$
DOFT	$\log_2 n$	0	$2 \log_2 n$	N	-	$\log_2 n(H + X)$
DHLHK	$\log_2 n$	$\log_2 n$	0	S	-	$(\log_2 n + 1)Ex$
DFT	n	0	$2n$ - 1	N	$(i\text{-}1)E$	iD

Tabela 4-7 Comparação dos protocolos descentralizados

n	Número de membros no grupo
i	Indíce do membro
I	Indice do tamanho
H	Resultado da função injectiva
X	Operação xor
E	Cifra
D	Decifra
Ex	Exponenciação
M	MAC

Tabela 4-8 Nomenclatura utilizada

4.1.4. Assinaturas digitais de grupo

Uma assinatura digital de grupo estende o conceito tradicional de assinatura digital a um conjunto de utilizadores. Num esquema deste tipo, os membros de um grupo podem assinar em nome do grupo, podendo as assinaturas serem verificadas utilizando uma chave pública de grupo. Após a assinatura de um documento ninguém, excepto o gestor do grupo, pode determinar quem o assinou realmente. As assinaturas de grupo devem ser desenhadas de modo a que não seja possível a nenhum membro do grupo falsificar uma assinatura, ao se fazer passar por outro membro do grupo.

As assinaturas digitais de grupo podem ser utilizadas, por exemplo, para validar listas de preços, comunicados ou contratos digitais em nome da companhia, sendo apenas necessário que se conheça uma chave pública, que permita verificar a validade do conteúdo. Desta forma esconde-se a estrutura interna da empresa, pois apenas o gestor do grupo tem a possibilidade de saber quem assinou o documento, dado que se garante que as assinaturas não podem ser forjadas pelos seus empregados.

Um esquema de assinaturas digitais de grupo é composto por cinco procedimentos base:

- *Setup* – tem como resultado a geração da chave pública e da chave secreta de administração (do conhecimento exclusivo do gestor do grupo), através da utilização de um algoritmo probabilístico.
- *Join* – protocolo interactivo entre o gestor e um potencial novo membro do grupo cujo resultado final consiste na produção de uma chave secreta por parte do novo membro, e de um certificado que o identifica como elemento efectivo do grupo.
- *Sign* – primitiva que recebe uma mensagem e uma chave secreta de um membro e retorna a assinatura sobre a mensagem, utilizando um algoritmo probabilístico.
- *Verify* – consiste num algoritmo que recebe uma mensagem, a sua assinatura (*digest* da mensagem cifrada com a chave privada do utilizador do grupo) e a chave pública do grupo e valida a veracidade da assinatura (decifrando-a com esta última).
- *Open* – um algoritmo que utilizando uma mensagem, a sua assinatura, e a chave secreta do gestor de grupo, determina a identidade do membro do grupo que assinou a mensagem.

Os protocolos de assinaturas digitais de grupos têm de garantir seis requisitos distintos de segurança, perante qualquer situação ou tentativa de ataque. São eles:

1. Impossibilidade de falsificação**:** Só os membros do grupo podem emitir assinaturas válidas em nome do grupo, ou seja, só os membros do grupo podem emitir assinaturas verificáveis pela chave pública do grupo.
2. Anonimato condicional do assinante**:** Apenas o gestor do grupo pode determinar a identidade do membro que assinou a mensagem. Há também a variante em que não existe um gestor de grupo, sendo deste modo impossível obter a identidade do assinante [26].

3. Impossibilidade de negação da identidade do assinante**:** O gestor do grupo pode sempre determinar a identidade do membro do grupo que emitiu a assinatura. Mais ainda, pode provar perante uma entidade terceira qual o membro que assinou a mensagem sem comprometer o anonimato desse membro em mensagens futuras ou passadas.
4. Impossibilidade de relacionamento das assinaturas**:** Determinar se duas assinaturas distintas foram emitidas pelo mesmo membro do grupo é computacionalmente inviável para todos, excepto para o gestor do grupo.
5. Segurança contra *Framing Attacks***:** Nenhum subconjunto de membros de um grupo (incluindo o gestor do grupo) pode assinar mensagem em nome de outro membro.
6. Resistência a Coligações**:** Nenhum subconjunto dos membros do grupo (incluindo o gestor do grupo) tem a capacidade de gerar e emitir assinaturas que não possam ser determinadas. Em particular, querem-se prevenir os ataques em que um subconjunto de membros se reúnem e geram assinaturas que podem ser verificas pelo procedimento *Verify* mas para as quais o procedimento *Open* falha em revelar qualquer membro do grupo.

A eficiência das assinaturas de grupo é medida pelos seguintes parâmetros:

1. O tamanho (numero de bits) da chave publica do grupo.
2. O tamanho (numero de bits) da assinatura de grupo numa mensagem.
3. A eficiência dos procedimentos *Sign*, *Verify*, *Setup*, *Open* e *Join*.

4.2. Resumo de casos de estudo

Vamos agora analisar três casos de estudo. Cada um tem as suas limitações e vantagens, e efectuam diferentes abordagens aos SCG e protocolos de geração e distribuição de chaves.

4.2.1. Secure Spread

O SecureSpread tem como objectivo estender o sistema de comunicação em grupo Spread [27] com serviços de segurança.

O Spread é um SCG de uso geral para redes locais ou alargadas. Disponibiliza a entrega de mensagens de forma fiável e ordenada (FIFO, causal, ordenação total), e um serviço de *membership*.

Este SCG consiste num servidor e numa biblioteca cliente interligada com a aplicação alvo, oferecendo um paradigma de comunicação onde um membro pode ser simultaneamente emissor e receptor.

Para garantir que os membros do grupo recebem o mesmo conjunto de mensagens entre dois eventos de estado do grupo, o Spread utiliza a semântica de *Extended Virtual Synchrony (EVS)*.

Por outro lado, o SecureSpread utiliza o pacote Flush [28] para lhe fornecer a semântica *View Synchrony*. A diferença entre estas duas semânticas é que na semântica EVS uma alteração ao estado do grupo ocorre sem intervenção do cliente, enquanto na *View Synchrony* o cliente tem que dar a sua permissão antes de uma nova visão do grupo ser aceite. Nesta semântica, quando ocorre uma alteração à *membership,* o SCG gera uma mensagem a indicar que a visão corrente do grupo está desactualizada e a pedir permissão para instalar a nova visão do grupo. Nesta fase o cliente ainda pode enviar e receber mensagens. No entanto, quando aceitar a permissão para instalar a nova visão de grupo, deixa de poder receber e enviar mensagens até que esta nova visão de grupo seja recebida por todos os elementos do grupo.

O SecureSpread também implementa a noção de *closed-group,* que consiste na propriedade em que só os membros do grupo podem efectuar operações relacionadas com este, tal como enviar mensagens para o grupo.

Para controlo de acessos o Spread utiliza um *framework* de autenticação [29] que suporta a implementação dos actuais sistemas standard de autenticação (PAM [31], SecureID [30], entre outros).

Para geração e distribuição de chaves o Spread foi integrado com o pacote CLIQUES 1.0. Este pacote baseia-se no uso do GDH para a geração e distribuição de chaves.

O SecureSpread utiliza ainda assinaturas digitais baseadas em RSA ou DSA para autenticar as mensagens durante as operações de IKA e AKA do GDH. As mensagens trocadas pelos membros do grupo são apenas cifradas, não tendo qualquer tipo de autenticação.

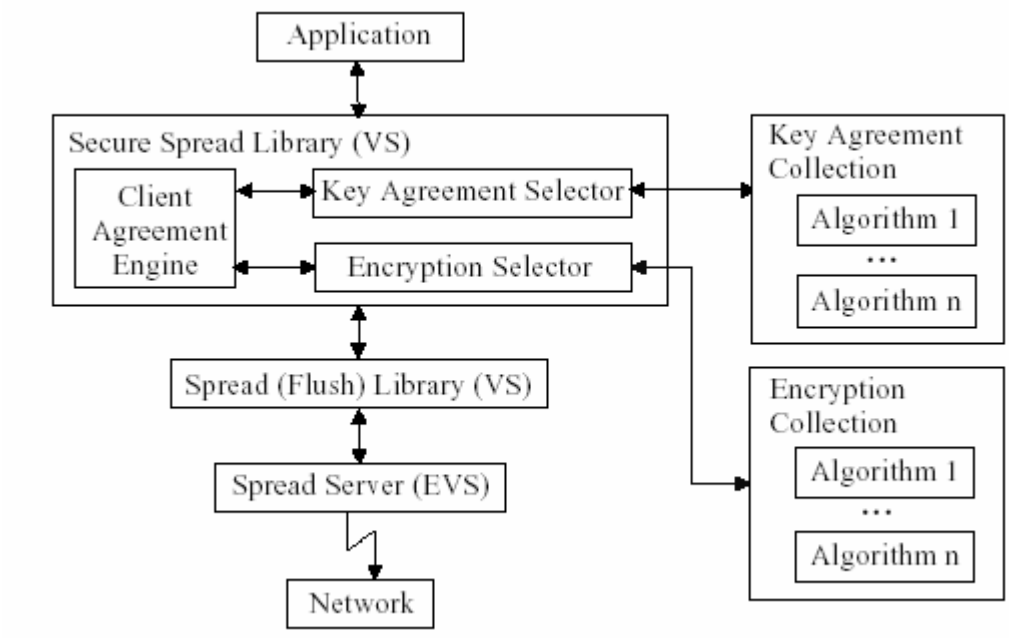

Figura 4-20 Arquitectura do SecureSpread

4.2.2. Segurança no sistema Ensemble

O Ensemble [34] é um SCG baseado no HORUS [41] que suporta entrega de dados de forma fiável e ordenada. Suporta *multicast* FIFO, comunicações ponto a ponto e, opcionalmente, *multicast* com ordenação total e detecção de falhas, entre outras propriedades.

O Ensemble, tal como o SecureSpread, consiste numa biblioteca que deve ser utilizada por aplicações alvo. Está dividido em camadas e cada aplicação utiliza a camada que precisar, fazendo assim o seu Ensemble específico.

O Ensemble utiliza uma *membership* do tipo *Virtual Synchrony* para manter a informação sobre membros do grupo e a chave do grupo.

Neste modelo existem dois tipos de mensagens: mensagens regulares e mensagens de *gossip*. Mensagens regulares são as mensagens normais de comunicação em grupo e as mensagens de *gossip* são geradas pelo Ensemble para comunicação dos vários componentes.

Regular	***Secure***

Top	
	Exchange
	Rekey
Gmp	
Top_Appl	
	Encrypt
RFifo	
Bottom	

*Router*s

Figura 4-21 As várias camadas do Ensemble.

A separação vertical apresentada na figura divide a extensão efectuada pelo Ensemble (zona *Secure*) ao HORUS (zona *Regular*). O *Top* e *Bottom* definem o início e fim do *stack*. O Gmp é o protocolo de *membership*, o Top_appl é a interface com a aplicação e RFifo é camada responsável pela entrega fiável dos dados. As camadas *Exchange* e *Rekey* estão relacionadas com a *membership* e, finalmente, a camada *Encrypt* é a responsável pela confidencialidade dos dados.

Para autenticar os membros o Ensemble utiliza o PGP. O sistema, por defeito, utiliza RC4 [35] para cifrar e para assinar as mensagens utiliza MD5 [36].

Com o objectivo de minimizar o custo de uma operação de *re-key*, este sistema utiliza a seguinte abordagem:

1. Inclui algoritmos rápidos de *re-key*.
2. O sistema efectua o *re-key* sempre de 24 em 24 horas.
3. A operação de *re-key* é inicializada pelo utilizador/aplicação. Desta forma, ele pode optar por uma maior *performance* e menor segurança, ou por uma menor *performance* e maior segurança.

Na camada mais baixa situa-se o módulo de autenticação do *router*. Os *router*s têm este nome devido a serem muito parecidos com os *routers* de IP, pois determinam como as mensagens devem ser enviadas para o processo de destino, e quando recebem mensagens determinam para que camada a mensagem se destina.

É esta camada a responsável pela assinatura das mensagens, em que cada *router* utiliza o MD5 para assinar e verificar as mensagens que envia e recebe. Os *router*s utilizam a chave de grupo para assinar as mensagens. Deste modo as mensagens trocadas pelos membros podem ser verificadas, mas as mensagens de *gossip* podem ser um problema, já que diferentes *router*s podem não usar a mesma chave de grupo.

Assim temos os seguintes casos:

- Seja *m* uma mensagem regular:
    1. Correctamente assinada: Passa para a próxima camada do stack.
    2. Incorrectamente assinada: É descartada já que pode vir que um diferente membro ou de alguém que tenta enviar mensagens para o grupo.

- Caso *m* seja uma mensagem de *gossip*:
    1. Correctamente assinada: Passa a próxima camada do stack.
    2. Incorrectamente assinada: é marcada como insegura e passa para o stack. A assinatura MD5 é ignorada. Como é possível que a mensagem no interior esteja assinada, através de assinaturas digitais, a camada *Exchange* irá tentar verificar a assinatura. Caso seja bem sucedida irá processar o conteúdo da mensagem *m*. Esta camada é a única que examina estas mensagens, todas as outras camadas ignoram as mensagens de *gossip* inseguras.

Como já foi referido, a camada de *Exchange* é a responsável pela negociação da nova chave de grupo e detectar partições no grupo. Para este efeito, o Ensemble elege automaticamente um líder do grupo. Este líder tem o protocolo *Heal* que é o responsável pela descoberta de partições no grupo, sendo enviadas periodicamente mensagens do tipo *IamAlive.* Quando forem recebidas mensagens de outro líder é iniciado o processo de *merge*.

Como os membros de um grupo só podem comunicar entre si quando todos os membros tiverem a mesma chave de grupo, a camada de *Exchange* utiliza mensagens inseguras de *gossip* para negociar uma nova chave de grupo. A ideia resume-se a que um dos grupos utilize a chave do outro grupo, sendo iniciada a sequência de *merge* pelo protocolo de *Heal* após ter sido acordada a nova chave do grupo.

Assim, a camada de *Exchange* funciona através da criação e reconhecimento de dois tipos de mensagens de *gossip*: mensagens para o processo *p* em que o nome do principal é R_p, e mensagens em que a visão do grupo tem a chave key_p:

- Id: Contem R_p e um *nonce*.
- *Ticket*: Contem dados para enviar de forma segura para um processo *q*. Este cabeçalho é criado através da cifra dos dados com a chave pública *q* e assinado com a chave privada de *p*.

Os seguintes eventos são aplicados às mensagens de *gossip* pelo processo *q*:

- Em cada mensagem de *gossip* adicionar o cabeçalho Id.
- Após receber um Id(R_p, $nonce_p$) – caso este esteja marcado como inseguro, *p* seja de confiança e Rq < Rp, então cria uma mensagem *Ticket* do tipo *gossip* para *p*. O *ticket* contem a chave key_q e o $nonce_p$ para provar que a mensagem é recente.
- Após receber o *Ticket* de *p* para o processo *q*, em que *p* é de confiança, autenticado e verificado o *nonce* – decifra-a para obter key_p. Se $key_p == key_q$, então ignora a mensagem caso contrário $new\_key := key_p$. Pede à camada para iniciar a alteração ao estado do grupo, já que a chave de grupo é parte constituinte do mesmo. Com o novo estado do grupo a chave de grupo irá ser *new_key*.

Quando um líder *q* recebe uma mensagem de líder remoto *p*, verifica se Id é inferior. Em caso afirmativo envia para *p* a chave key_q. O líder remoto autentica *q*, decifra a chave key_q e altera a visão de grupo para key_q. Daqui em diante todas as mensagens de *gossip* (já utilizando a nova chave) irão ser aceites, e os membros serão reunidos pela conclusão da operação *merge*.

Quando um processo falha, um líder (possivelmente novo) inicia a alteração ao estado do grupo. A chave do grupo não é alterada, já que o Ensemble acredita na honestidade dos membros, e que estes não divulgarão a chave de grupo a terceiros.

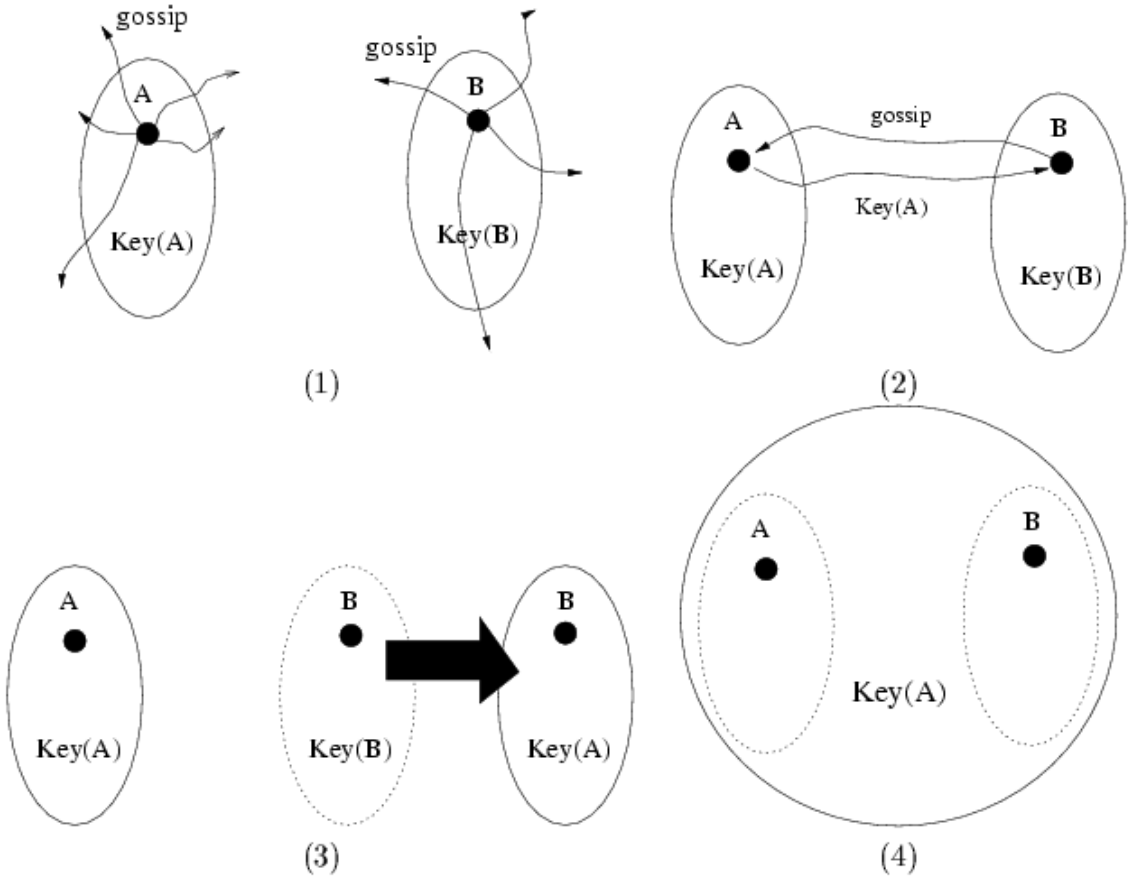

Figura 4-22 As várias etapas da operação de *Merge*. (1) Os líderes A e B enviam mensagens de gossip. (2) Líder A recebe a mensagem de B e envia a sua chave a B. (3) B altera a sua visão do grupo para a nova chave. (4) Os membros formam um novo grupo já que usam a mesma chave.

Este modelo também suporta a operação de *Re-key*, tendo sido desenvolvidos vários sistemas de *re-key* [42, 43]. Apresenta-se a seguir uma simples abordagem para a implementação de um destes protocolos. Ao contrário da camada de *Exchange* em que se usava a chave antiga para disseminar a nova chave, nesta camada não se reutiliza a chave.

O protocolo funciona nos seguintes passos:

- O líder escolhe uma nova chave, não relacionada com a antiga.
- Cifra, assina e envia a nova chave para cada membro utilizando um canal seguro (tipicamente um canal utilizando DH).
- Após receber a nova chave, cada membro envia uma confirmação ao líder.
- Quando o líder receber confirmação de todos os membros, inicia uma alteração ao estado do grupo.

Caso algum membro não confirme a nova chave então pode ter ficado particionado. Ocorre uma nova visão de grupo excluindo este membro e a antiga chave mantém-se. O líder é notificado de que a operação

de *re-key* falhou e pode tentar de novo. A segunda invocação da operação de *re-key* deve ser bem sucedida já que o membro em falha foi removido.

Em [44] foi desenvolvido um novo protocolo de *re-key* que tem como objectivo reduzir o número de canais seguros entre os membros. Com este novo protocolo consegue-se maior *performance* devido ao menor custo de gestão de vários canais seguros e à diminuição do custo de computação efectuada pelos membros.

Finalmente, na camada *Encrypt* é feita a cifra/decifra das mensagens *regulares* com a chave do grupo. As mensagens do Ensemble apenas são assinadas e não cifradas, já que não contêm informação secreta sobre os membros.

4.2.3. GMKA – Group Key Management Architecture

Este modelo é diferente do anterior já que define uma arquitectura e não "apenas" um protocolo de geração de chaves. No entanto é importante fazer uma análise das operações que esta arquitectura suporta e como funcionam essas operações.

É de notar que este modelo, por ser uma arquitectura, é passível de comparação com outros modelos SCG que integrem um protocolo de geração e distribuição de chaves, tal como o GDH.

Esta arquitectura está a ser desenvolvida por um grupo do IETF, o MSEC WG [25] e tem como objectivo o desenvolvimento de um *standard* para proporcionar segurança a comunicações de grupo, especialmente sobre o protocolo IP *multicast*.

Para esta arquitectura foi desenvolvido um protocolo de gestão de chaves de grupo em que o seu objectivo é proporcionar aos membros do grupo uma *security association* (SA) actualizada. Para alcançar este objectivo o GKMA utiliza os seguintes protocolos:

1. Protocolo de registo.

2. Protocolo de *re-key*.

O protocolo de registo é um protocolo ponto a ponto entre o controlador de grupo/servidor de chaves (GCKS) e um elemento que deseja entrar no grupo.

Neste protocolo o GCKS e o elemento autenticam-se mutuamente. Caso a autenticação seja bem sucedida e o membro seja autorizado a entrar no grupo, então o GCKS fornece-lhe a seguinte informação:

a) Informação suficiente para iniciar o protocolo *re-key*. Esta informação apenas é enviada caso seja exigida pela política de grupo.

b) Informação suficiente para iniciar o protocolo de segurança de dados. Esta informação apenas é enviada caso a politica de grupo implique a necessidade de iniciar o protocolo de segurança de dados durante o registo, em vez de, ou em conjunto, com o protocolo de *re-key*.

O protocolo de registo deve assegurar que a troca de informação entre o GCKS e o novo membro seja efectuada de forma autenticada e confidencial sob a forma de uma SA.

O protocolo de *re-key* é opcional, e como resultado o GCKS envia informação sobre novas chaves ao grupo. As mensagens de *Re-key* podem resultar de alterações ao estado do grupo, criação de novas chaves de cifra (TEKs) para um grupo em particular ou expiração de chaves. As mensagens de *Re-key* estão protegidas pela SA do protocolo de *Re-key* que foi iniciado durante o protocolo de registo, e incluem informação sobre como actualizar a SA do protocolo de *Re-key* e a SA do protocolo de dados. As mensagens de *Re-key* podem ser enviadas em *broadcast* para o grupo ou ponto a ponto para um membro do grupo.

A SA do protocolo *Re-key* inclui dados para autenticar a troca de mensagens. Estes dados podem assumir a forma de autenticação na fonte, em que os membros verificam que os dados de *Re-key* foram emitidos pelo GCKS, ou a forma de uma chave simétrica, utilizada para a troca de mensagens. Esta segunda forma só é aplicável quando for garantido que os membros nunca se poderão fazer passar pelo GCKS.

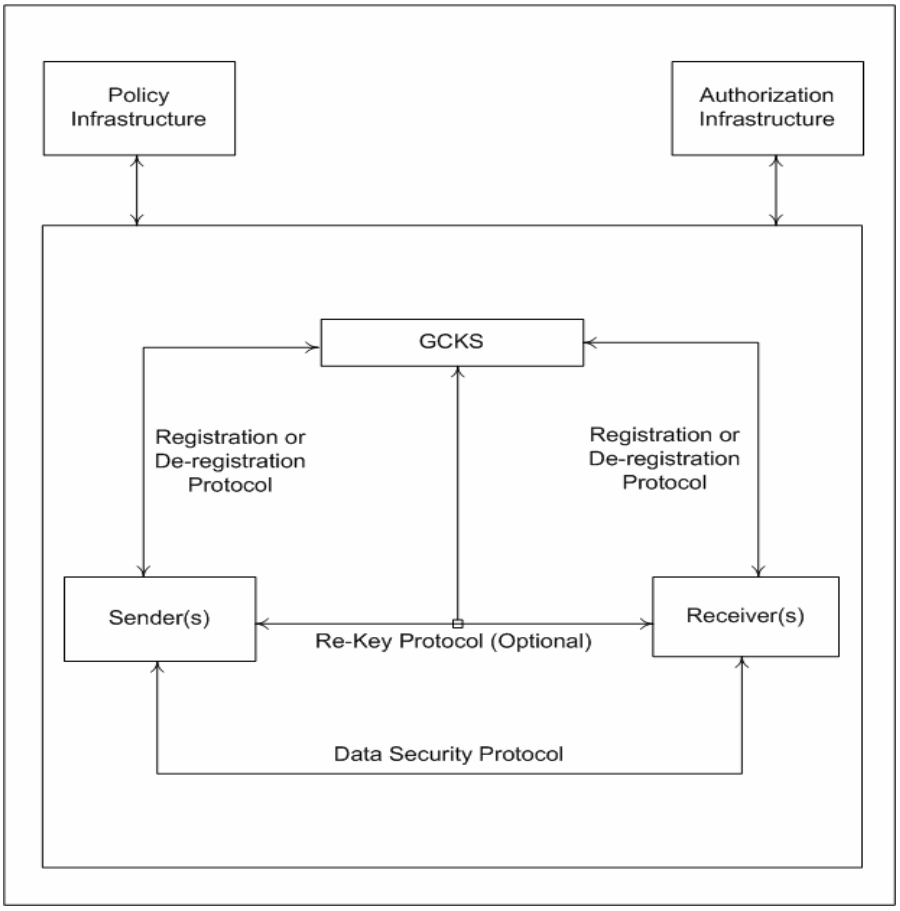

Figura 4-23 Esquema do GKMA

É facilmente compreensível que o protocolo de registo seja realizado nas operações de *Join* e criação do grupo, e que o protocolo opcional de *re-key* possa ser efectuado na outras operações: *Leave, Merge, Partition*.

Em resumo, cada membro do grupo, receptor ou emissor, utiliza o protocolo de registo para obter a autorização e autenticação num grupo específico, a sua politica e as suas chaves. Os dois tipos de chaves de grupo são as KEK (*key-encrypting key*) e TEK(*traffic-encrypting key*). O KEK pode ser uma simples chave que cifra o TEK, ou um vector de chaves de um algoritmo de chaves de grupo, que cifra o TEK e outros KEKs. O KEK é usado no protocolo *Re-key* e o TEK é usado no protocolo de dados para cifrar ficheiros e mensagens, ou seja, todo o tipo de dados enviados e recebidos pelo protocolo de dados.

O GCKS cria e envia os TEKs e KEKs para cada membro do grupo. É esta entidade lógica que efectua a autenticação e autorização dos membros de acordo com a política do grupo. O GCKS pode apresentar uma credencial ao membro do grupo, assinada pelo dono do grupo, para que esse membro possa verificar a autorização do GCKS.

Para melhor compreender o funcionamento desta arquitectura, vamos aprofundar a análise do funcionamento dos dois protocolos, Registo e *Re-key*.

O desenho do protocolo de Registo é flexível e tem como objectivo estabelecer um KEK ou múltiplos TEKs ou ambos KEK e TEKs, estando estes últimos associados ao protocolo de dados.

O protocolo de registo é protegido por uma SA simétrica e ponto a ponto entre o GCKS e cada membro. Este protocolo pode utilizar criptografia assimétrica para autenticar o controlador de grupo perante os membros do grupo e, opcionalmente, estabelecer o KEK.

As propriedades de desenho do protocolo de registo implicam restrições à segurança do mesmo: um protocolo que proporcione *perfect forward secrecy* (PFS) e protecção de identidade oferece mais segurança do que *performance*, enquanto um protocolo que complete o registo utilizando apenas uma ou duas mensagens implica maior *performance* e eficácia do que segurança.

Num protocolo que utilize uma ou duas mensagens, para se obter protecção contra repetições, geralmente utiliza-se uma estampilha temporal ou um número de sequência. O primeiro necessita de sincronismo dos relógios e o segundo requer uma tabela com as mensagens recebidas num certo espaço temporal.

Para protocolos com três mensagens durante o estabelecimento de uma sessão utiliza-se *Nonces* para evitar ataques de repetição na troca das mesmas. Em protocolos com seis mensagens (caso do ISAKMP), é possível obter protecção de identidade e evitar ataques de repetição.

O protocolo de *Re-key* tem como função o transporte das chaves e SAs entre o GCKS e os membros do grupo. O GCKS envia mensagens de *Re-key* para actualizar a *Re-key* SA, inicializar o data SA ou ambos. As mensagens de *Re-key* estão protegidas pela *Re-key* SA. O GCKS pode actualizar a *Re-key* SA quando o estado do grupo for alterado ou quando as chaves KEK/TEK expirarem. O protocolo de *Re-key* deve ter as seguintes propriedades:

- Assegurar que todos os membros recebem a informação sobre a *re-key* em tempo útil.
- Especificar mecanismos para as partes envolvidas contactarem o GCKS e re-sincronizarem quando as chaves já expiraram e não foram recebidas mensagens de actualização.
- Evitar problemas de implosão e assegurar fiabilidade na entrega das mensagens de *Re-key*.

Este protocolo é o responsável pela escalabilidade desta arquitectura, sendo os seus objectivos os seguintes:

- Sincronização da politica de grupo
- Proporcionar privacidade e autenticação (simétrica ou assimétrica).
- Proporcionar uma eficiente actualização das chaves após alterações no estado do grupo ou quando estas expiram.
- Opcionalmente, permitir uma entrega fiável de todas as mensagens de *Re-key*.
- Baixa latência e alta *performance* de utilização.
- Utilizar IP *Multicast* ou *multi-unicast*.

Para estes objectivos se concretizarem é necessário ter em conta cinco aspectos importantes no protocolo, que serão descritos a seguir:

- O formato das mensagens de *Re-key*
- A entrega fiável das mensagens de *Re-key*
- Implosão
- Incorporar GKMAs em mensagens de *Re-key*
- Intercomunicação entre GKMAs

As mensagens de *Re-key* contêm *Re-key*/Data SAs bem como KEKs e TEKs. Estas mensagens são confidenciais, autenticadas e resistentes a ataques de reenvio.

A entrega fiável de mensagens de *Re-key* é da responsabilidade do GCKS. Este tem que se assegurar que todos os membros têm as SA correntes de *Data* e *Re-key*. Deste modo o GCKS precisa de utilizar um algoritmo de *Re-key* confiável em que as mensagens sejam trocadas de forma fiável.

Existem duas dificuldades para se alcançar este objectivo: as mensagens que actualizam as chaves de grupo podem ser perdidas, ou não serem recebidas por membros que estejam desligados.

São propostas três categorias de soluções:

- Repetir a transmissão das mensagens *Re-key* (nos casos em as mensagens são um ou dois pacotes IP)
- Utilizar um protocolo de *multicast* fiável.
- O uso de *Forward Error Correction* (FEC) [33] para codificar as mensagens de *Re-key* (com NACKs como resposta)

A Implosão é um fenómeno de saturação e pode ter as seguintes causas: o processamento associado à sincronização das chaves com o GCKS, e a entrega fiável das mensagens de *Re-key*, que pode implicar a retransmissão das mesmas.

Como soluções temos: a supressão de feedback, a agregação de respostas, e permitir a cada membro contactar cada um dos servidores de registo quando estão dessincronizados.

Para poderem ser incorporados, os algoritmos de gestão de chaves de grupo (GKMAs) precisam de cumprir os seguintes requisitos:

- Colisão: Os membros e não membros não devem ter acesso a chaves indevidas.
- Controlo de acessos futuros: os membros não podem ter acesso ao conteúdo das mensagens após terem saído do grupo
- Controlo de acessos passados: os membros não podem decifrar mensagens anteriores à sua entrada.

Para a intercomunicação entre GKMAs é recomendado o uso da seguinte informação nas mensagens de registo:

- O nome do GKMA
- A versão do mesmo
- O número de chaves que contém
- Informação especifica da versão
- Informação sobre a chave:
    - ID da chave
    - Tempo de vida da chave
    - Chave de cifra
    - ID da chave cifrado (opcional)

Capitulo 5 - Um modelo para autenticação, controlo de acessos e distribuição de chaves em grupo

Passa-se a apresentar neste capítulo a proposta de um modelo genérico de integração de um serviço de segurança para um sistema de comunicação fiável em grupo. Este serviço é composto por vários subsistemas, a saber: sistema de autenticação em grupo, sistema de autorização e mecanismo de controlo de acessos em grupo e sistema de geração e distribuição de chaves de sessão para comunicação confidencial em grupo.

5.1. Abordagem inicial

Nos principais modelos existentes, HORUS, Ensemble, Spread, não é possível adicionar novos suportes de autenticação, sistemas de controlo de acessos ou sistemas de disseminação de chaves. Os sistemas actuais não permitem a configuração dos vários componentes a nível de grupo, ou seja, não existe o conceito de definição de um grupo pelos componentes utilizados para autenticação, controlo de acessos e geração e disseminação de chaves. É para lidar com estas dificuldades que se apresenta este modelo genérico de SCG seguro.

Tal como nos outros modelos, pretende-se satisfazer os seguintes requisitos de segurança:

- Confidencialidade
- Autenticidade
- Não repudio
- Integridade

A confidencialidade é garantida através da utilização de cifra simétrica e a partir de um mecanismo para distribuição e partilha dinâmica de chaves de cifra em grupo, que pressupõe um esquema de renegociação dinâmica de chaves com bases nos seguintes critérios:

- Eventos de *Membership*, nomeadamente entrada e saída de membros, partições e reunificações de grupos.
- Temporal, no fim do período de validade da chave.

- Pedido explicito por parte dos membros.

A propriedade autenticidade é assegurada durante duas fases distintas: no processo de entrada e na comunicação em grupo.

No processo de entrada no grupo é essencial que o principal se autentique perante o sistema. Esta autenticação é valida durante a sessão do principal no sistema e garante a sua identidade quando este se junta a um grupo. Tipicamente são utilizados sistemas de autenticação convencionais tais como *username/password* ou certificados digitais.

Para assegurar a autenticidade das mensagens utiliza-se assinaturas digitais ou HMAC. As assinaturas digitais podem ser utilizadas em nome individual ou em nome do grupo. As suas propriedades permitem garantir a autenticidade das mensagens, e a sua utilização permite garantir o requisito de não repudio das mesmas.

A integridade é garantida através do uso de funções de *hash* que estão presentes nas assinaturas digitais e no HMAC.

O uso do SCG permite construir componentes tolerantes a falhas, quer através da sua replicação, quer através da tolerância a falhas das comunicações entre os componentes.

O modelo consiste em quatro componentes principais, o *authentication service (AS)*, o *access control service (ACS)*, o *group comunication service (SCG)* e o *group key distribution center (GKDC)*.

O SCG é responsável pela coerência da visão de grupo e pela disseminação das mensagens para o mesmo, devendo ser fiável e resistente a falhas de comunicação.

O objectivo do AS é autenticar os indivíduos que desejam entrar no grupo, só após a validação dos utilizadores no AS é que é permitida a sua entrada. Este componente deve suportar várias políticas de autenticação.

O objectivo do ACS consiste no controlo de acessos dos membros. Quando um membro efectua alguma operação básica de um SCG, é verificada a sua legitimidade para efectuar tal operação.

Finalmente, o GKDC é responsável pela criação e disseminação de chaves de grupo. Este componente pode suportar diversos algoritmos de disseminação e criação de chaves, onde se destaca a forma centralizada e a criação de chaves de forma contributiva.

A partir destes componentes é possível definir um grupo seguro como um grupo que herda as propriedades de grupo de um SCG, como por exemplo, a *membership*, e que também utiliza um pacote de cifra simétrica em que a chave para cifra é criada dinamicamente. Além disso, utiliza um sistema de autenticação das mensagens, quer através da utilização de assinaturas digitais, quer através de HMAC. A utilização de politicas de acesso permite definir quais os membros que podem aceder ao grupo e quais as operações podem executar. Esta será a definição usada no modelo apresentado.

A autenticação dos principais que entram num grupo é baseada no princípio da validade da autenticação no sistema de comunicação segura em grupo, que tem de ser efectuada previamente.

A integração deste modelo nos sistemas actuais é resumida na seguinte figura:

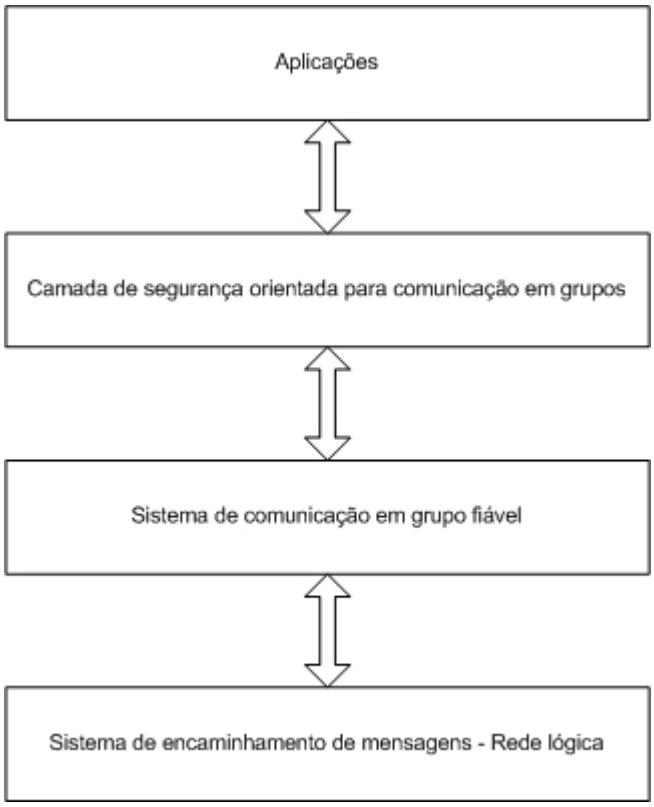

Figura 5-1 Integração do modelo nos sistemas existentes

O modelo está estruturado segundo uma lógica de camadas: primeiro aparece a camada aplicacional, onde residem as aplicações e integração com a API do modelo; a seguir a camada de segurança que permite obter um sistema de comunicação em grupo seguro, e por fim a camada que permite a comunicação em grupo. A última camada corresponde às infra-estruturas lógicas de comunicação.

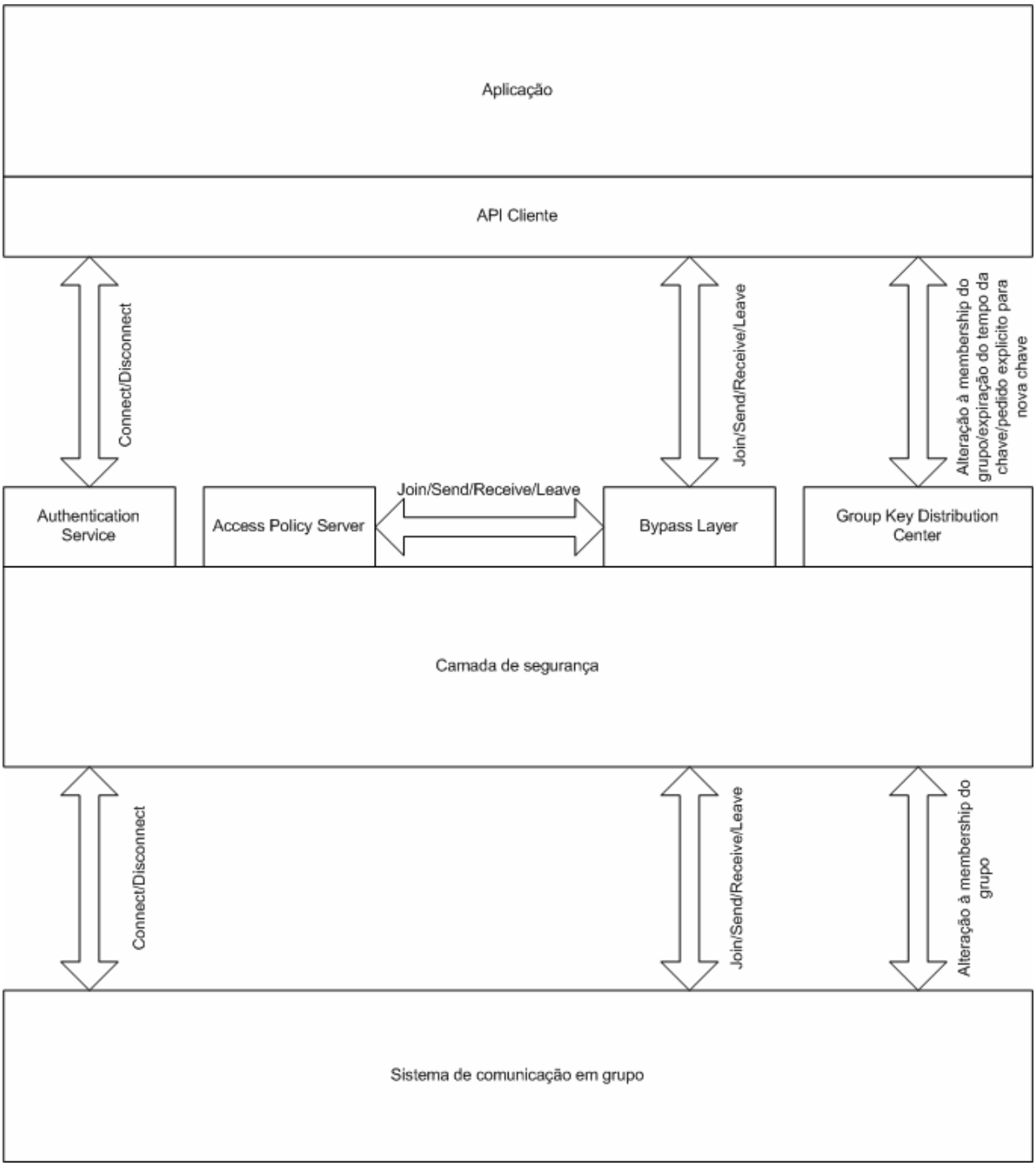

Figura 5-2 Estrutura do modelo apresentado e interacção com os seus componentes

Em detalhe temos a seguinte representação:

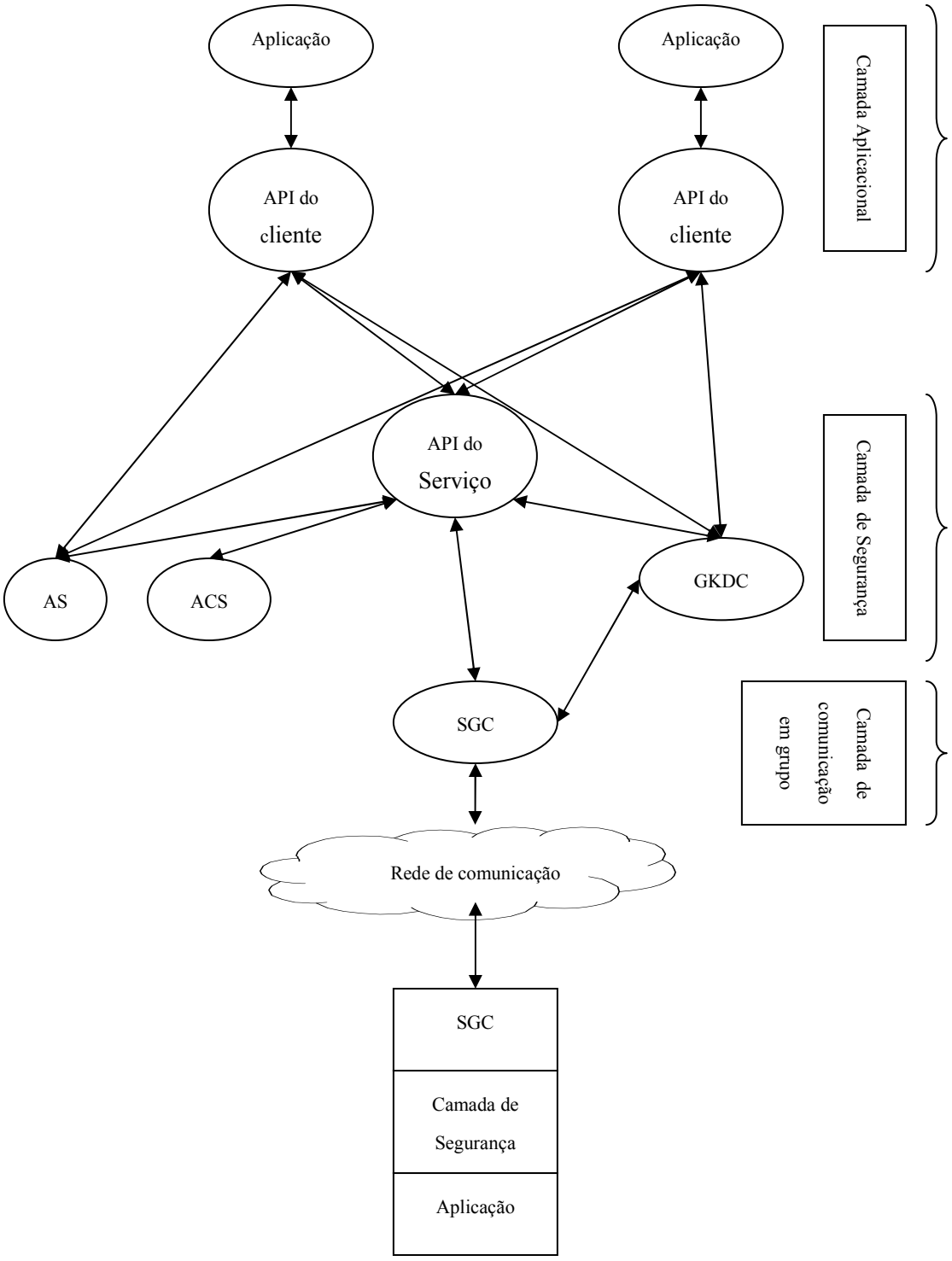

Figura 5-3 Esquema do modelo

Com o objectivo de modularidade, o modelo está dividido em duas APIs diferentes. A API cliente tem como objectivo tornar transparente para as aplicações a realização dos contactos, e o envio e recepção das mensagens trocadas entre os vários componentes. Esta API, além de comunicar com a API servidor, fala

com o AS para autenticar o utilizador, e com o GKDC para pedir as chaves necessárias para cifra e/ou assinaturas de grupo, bem como a implementação do protocolo de cifra e assinaturas de grupo a utilizar.

A API do servidor tem como objectivo abstrair a implementação dos vários componentes, fazendo com que a implementação dos mesmos seja independente. Deste modo obtém-se um modelo escalável e modular.

Todos os componentes apresentados são pontos de falha que acrescentam uma maior dificuldade de gestão e controlo da disponibilidade dos mesmos. Este modelo não implementa nenhum sistema de alta disponibilidade e detecção de falhas no que diz respeito aos componentes, cabe a cada um deles garantir a existência destas duas propriedades.

Para efeitos de implementação, deve-se considerar o uso do SCG como forma de comunicação entre todos os componentes. Desta forma é possível utilizar as propriedades de alta disponibilidade das comunicações e detecção de falhas de um SCG, podendo ser implementadas soluções de alta disponibilidade de todos os componentes apresentados.

5.1.1. Componente SCG

Este componente é a ponte de ligação entre os vários membros do grupo. Deve suportar vários algoritmos de entrega de mensagens e o conceito de *membership*. Tal como já foi referido o SCG deve suportar as seguintes operações:

Join – Entrada de um membro do grupo

Leave – Saída de um membro do grupo

Send – Envio de mensagens para o grupo

Receive – Recepção das mensagens do grupo

Deve, ainda, ser resistente a falhas, especialmente, partição dos membros do grupo (*partition*) e a sua reunificação (*merge*).

Neste modelo, o algoritmo base de entrega de mensagens é o FIFO.

Dado que os objectivos dos SCG são comuns a todos eles, as suas arquitecturas são semelhantes. Assim, estas arquitecturas não serão novamente apresentadas aqui, pois já foram objecto de estudo em capítulos anteriores.

5.1.2. Componente de autenticação (AS)

É neste componente que a autenticação se irá efectuar. O seu objectivo principal consiste em comunicar, aos utilizadores e ao servidor do modelo, se um determinado utilizador tem ou não permissão para entrar no sistema.

Para cumprir o objectivo de suportar diversos métodos de autenticação, este componente é genérico e baseia-se em funções genéricas, que os diferentes métodos de autenticação terão de implementar. A autenticação é válida durante toda a ligação ao servidor.

É importante que este componente seja fácil de implementar e que a integração de sistemas já existentes, como por exemplo o PAM (*Pluggable Authentication Module*), seja simples.

Assim, a arquitectura do AS tem como requisitos a simplicidade e a possibilidade de expansão.

5.1.2.1. Arquitectura do AS

O AS consiste em duas APIs de autenticação distintas: uma é a API chamada pelo cliente, neste caso, membro do grupo; a outra é a API chamada pelo modelo, para requerer a autenticação do cliente.

Local de execução	***Operação/Função***	***Parâmetros***	***Resultado***
Servidor	Start_AS	-	Sucesso ou código de erro
Servidor	Validate_User	Nome do utilizador	Sucesso ou código de erro
Cliente	Validate	-	Sucesso ou código de erro

Tabela 5-1 Descrição das APIs do AS

A validade da autenticação do utilizador é mantida durante a ligação ao SCG seguro. Esta validação é efectuada durante a ligação ao sistema e pode envolver mais de um servidor de AS. Cada aplicação irá implementar o seu AS conforme as suas necessidades.

O AS tem duas funções importantes:

- Uma das funções é solicitada pelo sistema, para questionar a autenticidade de um utilizador. Esta função implica a existência de troca de informação com o cliente, por forma a permitir a autenticação.
- A outra função é solicitada pelo utilizador e serve para contactar o servidor do AS, com o objectivo de trocar informação e executar o protocolo de autenticação.

5.1.3. Componente de controlo de acessos

Um SCG seguro não pode depender apenas da autenticação dos seus utilizadores para garantir o correcto funcionamento do grupo e dos seus membros. Um sistema de controlo de acessos é necessário para garantir que apenas certos membros possam efectuar operações num grupo.

O objectivo deste componente é controlar as acções de um membro num grupo. Através do uso de politicas é possível especificar quais as acções que um membro pode executar durante a sua existência no grupo, sendo as acções susceptíveis de controlo as operações básicas de um grupo (*Join, Leave, Receive, Multicast_Send, P2P_Send*).

O processo de geração e controlo das políticas de acesso está fora do âmbito da tese, devendo ser um processo paralelo ao modelo apresentado.

5.1.3.1. Arquitectura do ACS

Um ACS com o objectivo de controlar operações de membros em grupos requer politicas de utilização diferenciadas por grupo, isto é, um membro pode num determinado grupo enviar mensagens, enquanto noutro pode não ter essa mesma permissão. O objectivo de se ter esta granularidade no controlo de acessos resulta da possibilidade de integração de mais aplicações, e de um maior controlo no acesso de membros por parte dessas mesmas aplicações.

Um grupo só pode ter um sistema de controlo de acessos associado num dado tempo, no entanto este sistema pode ser alterado em tempo de execução, podendo ser parado ou haver uma alteração das suas políticas.

Enquanto um grupo só pode ter um ACS, um ACS pode controlar os acessos em diferentes grupos, com diferentes politicas em cada um.

Neste componente são efectuadas interacções com o servidor do modelo apresentado, onde para cada acção executada por um membro de um grupo, é efectuada a verificação da permissão correspondente. Caso não seja possível contactar o ACS, assume-se que o membro não tem permissão para executar a acção.

Existem duas verificações que, devido à sua complexidade na implementação e execução, poderão ser consideradas desapropriadas: a operação de verificação de um membro poder receber mensagens e a operação de um membro poder sair do grupo. Como é óbvio, é impossível ao modelo obrigar a que um membro não saia de um grupo de livre vontade ou controlar a saída inesperada de um membro. No entanto esta função poderá servir de registo de saídas de membros, uma forma de controlo e análise de desempenho.

Quanto à operação de verificação na recepção de uma mensagem de um membro, esta não deve depender apenas da verificação para garantir que o membro não recebe a mensagem, já que o SCG deve suportar alguma forma de disseminação de mensagens excluindo certos membros, sendo estes, por exemplo, dados em forma de lista. É de notar que esta operação será bastante pesada já que para todos os membros do grupo será necessário verificar a sua permissão.

Numa fase inicial deste modelo as duas operações referidas anteriormente não irão ser verificadas devido à sua complexidade e agravamento do desempenho do modelo.

Operação/Função	***Parâmetros***	***Resultado***
Start_ACS	-	Sucesso ou código de erro
User_can_Join	Nome do utilizador, Nome do grupo	Sucesso ou código de erro
User_can_Leave	Nome do utilizador, Nome do grupo	Sucesso ou código de erro
User_can_Receive	Nome do utilizador, Nome do grupo	Sucesso ou código de erro
User_can_Multicast_Send	Nome do utilizador, Nome do grupo	Sucesso ou código de erro
User_can_P2P_Send	Nome do utilizador, Nome do grupo	Sucesso ou código de erro
Add_ACS_2_Group	Nome do grupo	Sucesso ou código de erro
Remove_ACS_from_Group	Nome do grupo	Sucesso ou código de erro
Add_ACS_Policy	Nome do grupo, Policy	Sucesso ou código de erro
Update_ACS_Policy	Nome do grupo, Policy	Sucesso ou código de erro
Stop_ACS	-	Sucesso ou código de erro

Tabela 5-2 Descrição da API do ACS

5.1.4. Componente de geração e distribuição de chaves

Este módulo é o responsável pela geração e distribuição de todas as chaves a utilizar pelo grupo, sejam elas chaves para cifrar ou chaves para assinar os dados.

Tem como objectivo suportar diversos modelos de geração, gestão e distribuição de chaves.

Idealmente a chave de sessão deverá ser uma chave contribuída por todos os elementos e a assinatura digital deverá ser de grupo, caso em que existem várias chaves privadas para uma chave pública.

Existe no máximo um GKDC por grupo, ficando a cargo da implementação do GKDC a existência de replicação, para garantir alta disponibilidade e tolerância a falhas.

No GKDC a geração de chaves de grupo pode ser feita de forma centralizada ou contributiva. Por isso, o GKDC deve pertencer ao grupo, de forma a poder tirar partido das propriedades do SCG para a geração, disseminação das chaves e envio das mesmas para o grupo.

A geração de chaves depende da política que se deseja implementar. Por exemplo, caso se deseje uma aplicação com *performance*s óptimas pode-se utilizar sempre a mesma chave durante a entrada e saída de membros, sendo gerada e distribuída conforme o *key refresh* estabelecido. Caso se deseje uma aplicação mais segura pode utilizar-se uma politica de geração de uma nova chave apenas na saída de novos membros do grupo. Como o GKDC pertence ao grupo, tem conhecimento de qual o membro que saiu, e pode assim distribuir a nova chave utilizando o SCG, ou utilizar uma politica de distribuição ponto a ponto.

A política de distribuição de chaves está directamente implicada com o tipo de geração da chave, isto é: caso se utilize um método de geração contributário então a sua geração e distribuição poderá ser efectuada utilizando o SCG; se for utilizado um método de geração da chave centralizado, então a sua distribuição deverá ser ponto a ponto.

Cabe ao GDKC fornecer a implementação do sistema de cifra a utilizar. Por exemplo, caso se deseje utilizar DES com uma chave contributiva, deverá existir essa implementação. Esta implementação pode ser construída de uma forma tão simples como utilizar um protocolo de cifra já existente e apenas lidar com a geração e distribuição de chaves de grupo.

Caso não seja possível contactar o GKDC deve-se utilizar a chave de sessão que já era do conhecimento dos utilizadores. Em caso de entrada de um novo membro, este ficará inactivo no grupo até à disseminação da próxima chave de sessão.

O GKDC está dividido em dois sub componentes, o sub componente que interage com o servidor do modelo e o sub componente que interage com os utilizadores.

Operação/Função	***Parâmetros***	***Resultado***
Start_GKDC	-	Sucesso ou código de erro
Add_GKDC_2_group	Nome do grupo	Sucesso ou código de erro
Remove_GKDC_from_group	Nome do grupo	Sucesso ou código de erro
Add_GKDC_group_policy	Nome do grupo, Policy	Sucesso ou código de erro
Update_GKDC_group_policy	Nome do grupo, Policy	Sucesso ou código de erro
Stop_GKDC	-	Sucesso ou código de erro

Tabela 5-3 Descrição da API servidor do GKDC

O sub componente que interage com o servidor do modelo consiste na configuração do GKDC a nível de politicas (como por exemplo a especificação do *key refresh*), e da gestão da afectação dos GKDCs (para efectuarem o controlo da geração e disseminação de chaves num grupo).

O sub componente que interage com os utilizadores tem como objectivo desenvolver as implementações dos diferentes algoritmos de cifra e assinatura de chaves, bem como obter as chaves actuais de cifra e assinatura digital das mensagens.

A implementação do sistema de assinaturas deve ter no mínimo as funções de *Init, Sign* e *Verify*. A implementação do sistema de cifra deve ter no mínimo as funções de Init, *Encrypt* e *Decrypt*.

Além das óbvias implementações da operação de cifra, decifra, assinatura e verificação de assinaturas digitais cabe à implementação dos algoritmos de cifra e assinatura lidar com os eventos habituais de um grupo, isto é, a entrada, saída, partição e reunificação de membros.

Operação/Função	***Parâmetros***	***Resultado***
Get_GKDC_Encrypt_Implementation	Nome da implementação	Sucesso ou código de erro
Get_GKDC_Sign_Implementation	Nome da implementação	Sucesso ou código de erro
Get_GKDC_Group_Key	Nome do grupo	Sucesso ou código de erro
Get_GKDC_Group_Sign_Key	Nome do grupo	Sucesso ou código de erro

Tabela 5-4 Descrição da API do cliente do GKDC

5.2. Aproximação à concretização do modelo

Nesta secção descreve-se o desenho do servidor do modelo. O servidor do modelo tem como objectivo interligar os vários componentes e proporcionar uma API para o desenvolvimento de aplicações.

O servidor, tal como o GKDC, está dividido em dois sub componentes. A API para o desenvolvimento de aplicações, ou seja a interface de ligação com o utilizador final e uma outra API que tem como objectivo registar novos componentes de autenticação, controlo de acessos, sistemas de geração e disseminação de chaves e interligação destes mesmos componentes.

A API do cliente fornece as funções necessárias para o desenvolvimento de aplicações que utilizem este modelo, e como tal fornece as funções necessárias para a ligação ao servidor, criação de grupos, entradas em grupos, saídas de grupos, envio de mensagens para o grupo e recepção de mensagens de um grupo.

Fornece ainda quatro funções que ajudam na criação de grupos e autenticação nos mesmos; uma função que permite obter os sistemas de autenticação que o modelo suporta, para que um utilizador possa aceder ao SCG seguro, uma outra função que permite obter os vários tipos de ACS que existem, para controlo de acessos no grupo, a terceira função tem um objectivo semelhante à anterior já que permite obter a lista de GKDC que existem para uso na gestão das chaves num grupo. Estas duas últimas funções servem para decidir qual irá ser o ACS e GKDC a utilizar durante a criação do grupo e finalmente uma função que permite saber qual é o GKDC que um grupo está a utilizar.

Operação/Função	***Parâmetros***	***Resultado devolvido***
Connect	Nome do Servidor	Sucesso ou código de erro
Disconnect	-	Sucesso ou código de erro
Create	Nome do Grupo; esquema de controlo de acessos que o grupo irá utilizar; algoritmo de geração e distribuição a utilizar no grupo	Sucesso ou código de erro
Join	Nome do grupo	Sucesso ou código de erro
Leave	Nome do grupo	Sucesso ou código de erro
Multicast_Send	Nome do grupo; dados; Tipo de fiabilidade	Sucesso ou código de erro
P2P_Send	Nome do grupo, nome do receptor, dados	Sucesso ou código de erro
Receive	Nome do grupo	Estrutura com os dados e número de bytes recebidos ou código de erro
Get_Authentication_List	-	Lista com os esquemas de autenticação suportados pelo modelo ou código de erro
Get_ACS_List	-	Lista com os ACS suportados pelo modelo ou código de erro
Get_GKDC_Types	-	Lista com os GKDC suportados pelo modelo ou código de erro
Get_Group_GKDC_Type	Nome do grupo	O GKDC suportado pelo grupo ou código de erro

Tabela 5-5 Descrição da API do cliente do modelo

O outro sub componente consiste no registo de implementações dos componentes de AS, ACS, GKDC e interligação dos vários componentes. É este o componente responsável pela gestão do grupo versus os componentes de ACS, GKDC e a abstracção da API do cliente do modelo da interligação dos vários componentes.

Operação/Função	***Parâmetros***	***Resultado devolvido***
Register_Authentication_Module	Módulo de autenticação	Sucesso ou o código do erro
Register_ACS_Module	Módulo de controlo de acessos	Sucesso ou o código do erro
Register_GKDC_Module	Módulo de GKDC	Sucesso ou o código do erro

Tabela 5-6 Descrição da API do servidor do modelo

5.2.1. Tipos de mensagens

Existem três tipos de mensagens trocadas entre a API do cliente e os diferentes componentes do modelo: mensagens de autenticação, mensagens de trocas de chaves e mensagens de dados.

As mensagens de autenticação são as mensagens trocadas entre o indivíduo e o AS. Estas mensagens são utilizadas durante a operação de *Connect*.

As mensagens de trocas de chaves são utilizadas conforme a politica utilizada pelo GKDC.

As mensagens de dados são utilizadas em todas as operações que um utilizador pode efectuar.

5.2.1.1. Mensagens de autenticação

As mensagens de autenticação têm o seguinte formato:

{Identificador do utilizador, Dados específicos à implementação do AS, Estampilha Temporal, Autenticação da Mensagem}.

O Identificador do utilizador é o nome com que o utilizador se irá autenticar no AS.

Os dados relativos à implementação do AS representam os dados trocados entre o AS e a API do cliente. Por exemplo no sistema de autenticação *User/Password*, este campo conterá a password.

Estampilha Temporal é uma estampilha de Lambert com o objectivo de evitar *replaying* da mensagem.

Autenticação da Mensagem consiste ou numa assinatura digital da mensagem (neste caso o nome do utilizador deverá ser igual ao do certificado) ou no uso de HMAC. Tem como objectivo garantir a autenticidade da mensagem.

5.2.1.2. Mensagens de dados

As mensagens de dados têm o seguinte formato:

{ID privado, Tipo de operação, Nome do grupo, Dados específicos à operação, Estampilha Temporal, Autenticação da Mensagem}.

O ID privado é um identificador fornecido pelo servidor e que permite identificar a ligação.

Tipo de operação é um identificador que determina qual é operação que a mensagem se refere.

Nome do grupo é o identificador do nome do grupo utilizado no SCG.

Dados específicos à operação são os dados necessários para a execução da operação especificada. Por exemplo no caso da operação de Multicast_Send consiste nos dados a enviar para o grupo.

Estampilha Temporal é uma estampilha de Lambert com o objectivo de evitar *replaying* da mensagem.

Autenticação da Mensagem consiste ou numa assinatura digital da mensagem (neste caso o nome do utilizador deverá ser igual ao do certificado) ou no uso de HMAC. Tem como objectivo garantir a autenticidade da mensagem.

5.2.1.3. Mensagens de geração e distribuição de chaves

As mensagens de geração e distribuição de chaves têm o seguinte formato:

{Identificador do utilizador, Dados específicos à implementação do GKDC, Estampilha Temporal, Autenticação da Mensagem}.

O Identificador do utilizador é o nome com que o utilizador se irá autenticar no GKDC.

Os dados relativos à implementação do GKDC representam os dados trocados entre o GKDC e a API do cliente. Por exemplo no sistema de geração de chaves distribuídas, este campo conterá a computação relativa a este membro.

Estampilha Temporal é uma estampilha de Lambert com o objectivo de evitar *replaying* da mensagem.

Autenticação da Mensagem consiste ou numa assinatura digital da mensagem (neste caso o nome do utilizador deverá ser igual ao do certificado) ou no uso de HMAC. Tem como objectivo garantir a autenticidade da mensagem.

5.3. Interacções entre os componentes do modelo

Um principal percorre um conjunto de estados que correspondem às suas acções. O conjunto de estados de um principal está definido na seguinte figura:

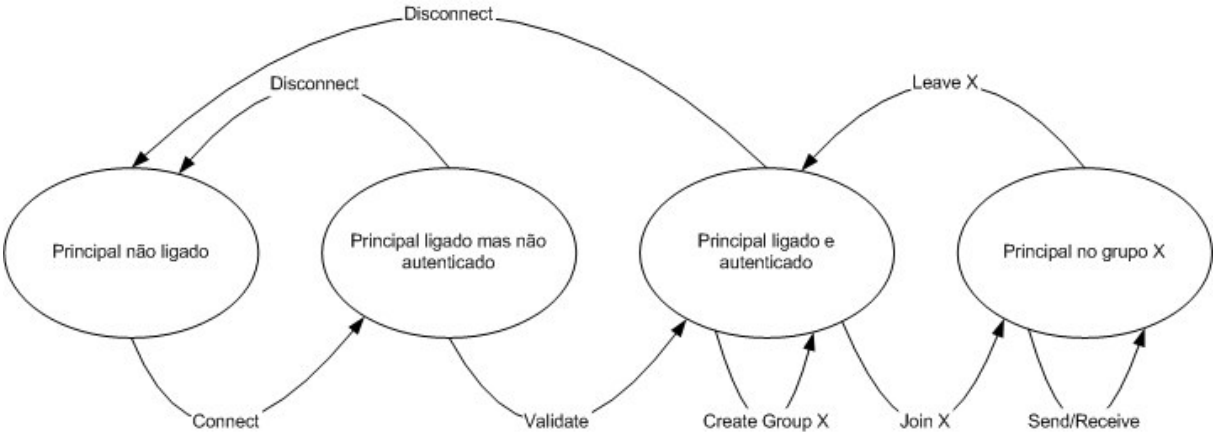

Figura 5-4 Estados de um principal

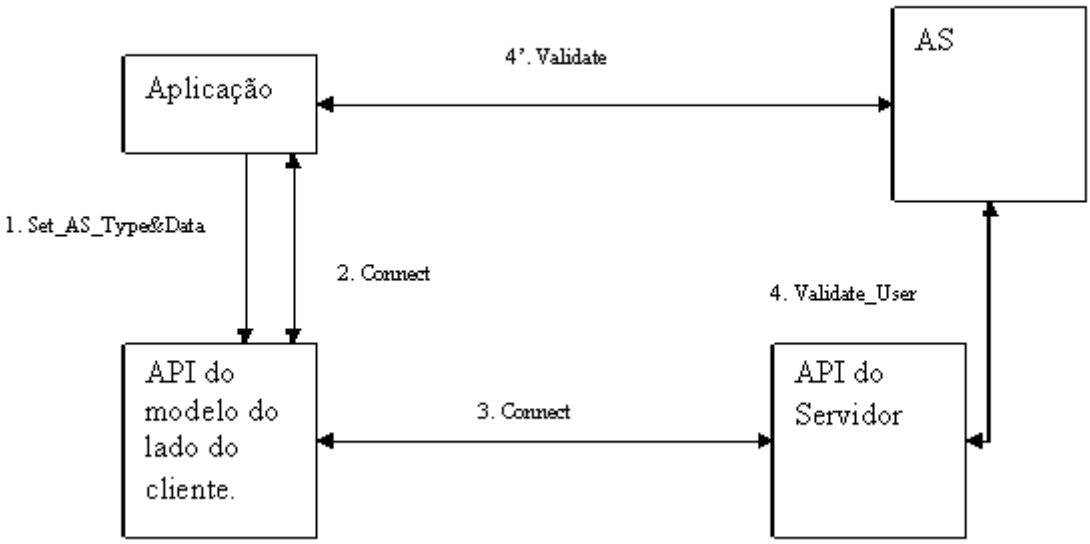

Figura 5-5 Operação de Connect ao servidor

A primeira iteração existe durante a operação de *connect* ao servidor. Nesta fase é pedido para o utilizador se autenticar perante o AS.

Durante a operação de inicialização é efectuado o registo dos vários tipos de GKDC que um grupo pode ter. Por exemplo podemos ter um GKDC centralizado que dissemina chaves de grupo simétricas, um GKDC em que as chaves de grupo são contribuídas por todos os elementos do grupo, etc.

Operação *create*:

Antes da operação *create* cabe ao utilizador seleccionar qual o tipo ou tipos de GKDC que deseja para a criação do grupo bem como o tipo de ACS a utilizar no grupo.

Durante a operação de *create*, o modelo inicia um novo GKDC do tipo seleccionado, que opcionalmente executa a operação de *Join* ao grupo.

Também nesta fase, o utilizador decide qual é o modelo de ACS que o grupo irá utilizar.

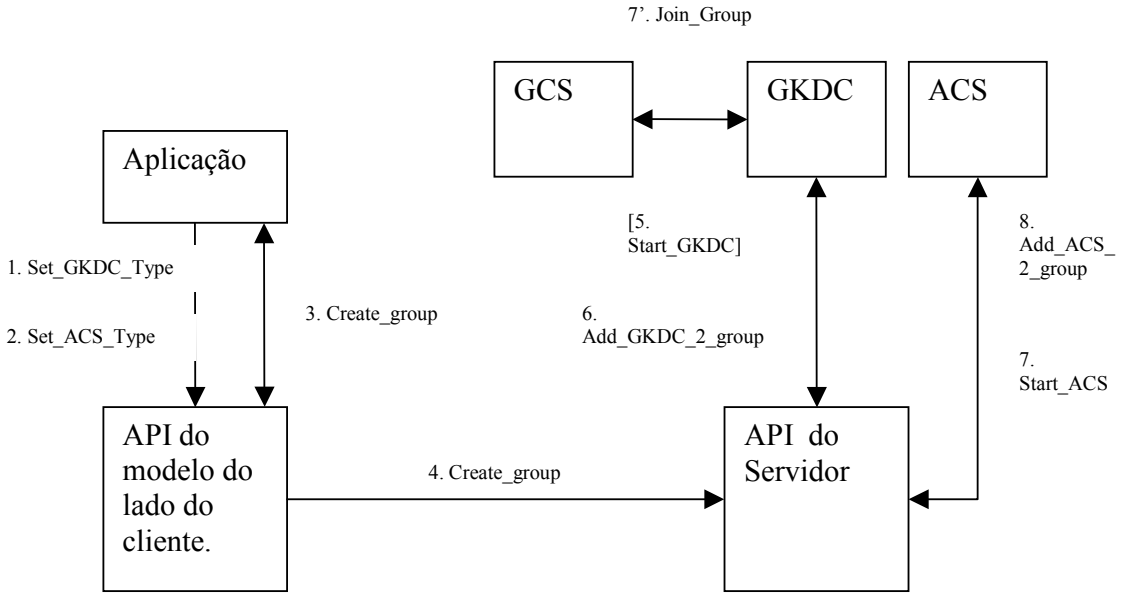

Figura 5-6 Operação de criação de um grupo

Operação *Join*:

Esta operação é efectuada em três fases:

1ª fase - contacto com o API do servidor e efectuado o controlo de acesso no ACS

2ª fase – pedido de *join* ao SCG

3ª fase - contacto com o GKDC.

Na primeira fase é efectuada a ligação ao servidor e efectuado o controlo de acessos no ACS.

Aqui começa a segunda fase, em que a API do servidor inicia o pedido *join* ao SCG. Por seu lado o SCG permite a entrada do novo membro no grupo, alterando a sua *membership*.

Na terceira e última fase, a API do cliente contacta o GKDC, apresentando o *ticket* da API do servidor e pedindo a implementação do sistema de cifra a utilizar, da chave do grupo. Opcionalmente, pode também pedir a implementação e a chaves para autenticar as suas mensagens.

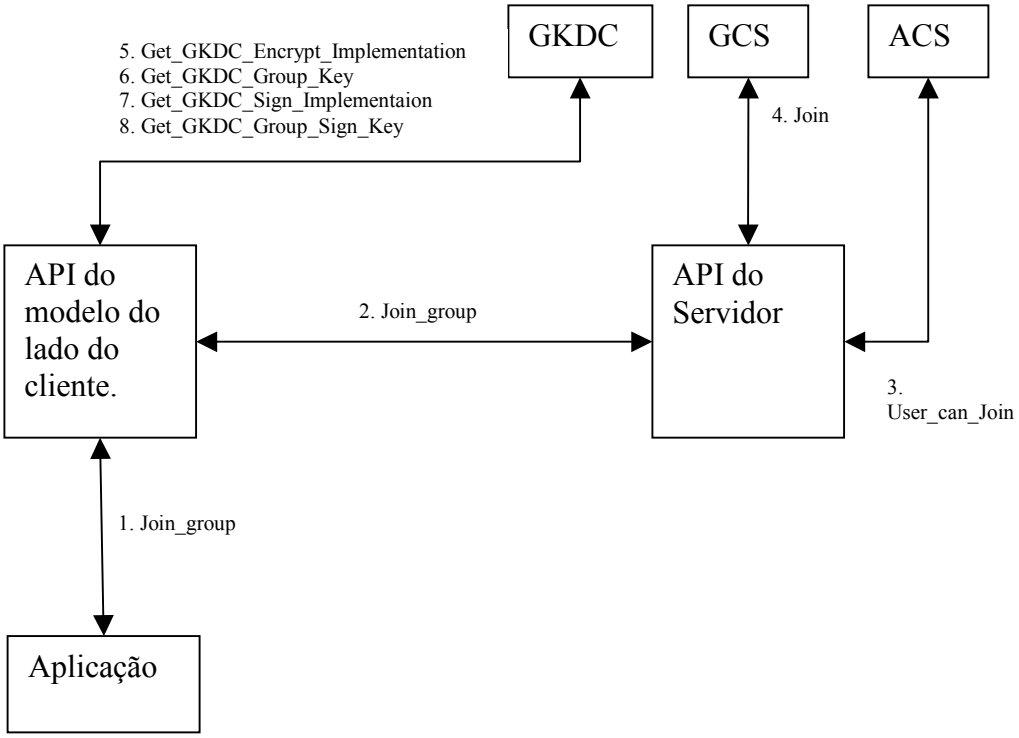

Figura 5-7 Operação de Join a um grupo

Operação *Leave*:

Após a actualização da *membership* por parte do SCG, esta operação obriga à geração da nova chave de grupo e sua disseminação.

Esta operação é a principal motivação para que o GKDC pertença ao grupo. Com a actualização da *membership* por parte do SCG, o GKDC tem a confirmação da saída dos membros do grupo.

Operação *Send*:

Pode ser considerada simples e implica a cifra da mensagem. Dependendo da implementação, podemos ter também um *message digest* e uma assinatura sobre a mesma.

Operação *Receive*:

Deve fazer a verificação da assinatura, caso exista, e decifrar a mensagem.

5.4. Comparação do modelo com Ensemble e SecureSpread

O SecureSpread utiliza o *framework* de autenticação e controlo de acessos do Spread. O pacote Spread de base fornece duas implementações para autenticação, uma baseada em *user/password* e a outra em IPs. No entanto não têm nenhuma implementação de um sistema de controlo de acessos. Como já foi referido utiliza o pacote CLIQUES, nomeadamente o GDH para geração e distribuição de chaves de grupo. Usa essas mesmas chaves com o protocolo de cifra Blowfish. Utiliza assinaturas digitais (RSA/DSA) durante a fase de geração e distribuição de chaves de grupo. Estas chaves são actualizadas com base numa politica *Membership-sensitive* [45].

O Ensemble deriva do Horus e como tal o seu SCG tem as mesmas propriedades do Horus, isto é resistente a partições de rede. Como sistema de autenticação utiliza o PGP apesar de ter uma interface com o Kerberos. Usa ACLs para controlo de acessos, é de notar que durante a fase de *merge* apenas é validado a entrada dos controladores de grupo entre si e não de todos os membros. Utiliza chaves simétricas centralizadas e como protocolo de cifra utiliza ou DES ou IDEIA ou RC4, aproveita a chave de grupo para utilizar como semente para assinar mensagens utilizando HMAC. A alteração à chave do grupo é efectuada numa politica *Time-sensitive* ou a pedido do utilizador.

O modelo apresentado suporta vários modelos de autenticação, controlo de acessos, protocolo de geração e distribuição de chaves bem como vários sistemas de cifra. Uma chave é actualizada conforme a politica imposta. O modelo é resistente a falhas desde que o SCG também o seja e cabe a cada componente manter a disponibilidade e integridade do seu serviço. Por exemplo o sistema de autenticação pode ser visto como um cliente deste mesmo modelo utilizando as propriedades do SCG para comunicar com várias réplicas do serviço.

Em seguida apresenta-se um quadro com as diferentes soluções apresentadas por estes modelos:

	SecureSpread	*Ensemble*	*Modelo apresentado*
Autenticação dos membros	Independente.	Utiliza PGP	Independente.
Controlo de Acessos	Independente.	Utiliza ACLs.	Independente.
Sistema de cifra	Blowfish.	RC4, DES, IDEIA.	Independente.
Geração das chaves de cifra	Contributivas.	Centralizadas.	Independente.
Autenticidade das mensagens	RSA/DSA apenas durante a geração da chave de grupo.	HMAC usando como semente a chave do grupo	Independente.
Re-keying	Membership	Time	Independente.
Tolerância a falhas	SCG resiste a partições de rede	SCG resiste a partições de rede	SCG resiste a partições de rede. Cabe aos componentes implementar alta disponibilidade.

Tabela 5-7 Resumo da comparação entre SecureSpread, Ensemble e o modelo apresentado

Capitulo 6 – Protótipo de Implementação

Neste capítulo será descrito o protótipo de implementação criado com base no modelo apresentado. Na primeira parte deste capítulo será efectuada uma breve descrição de quais as componentes implementadas, efectuando o seu mapeamento para o modelo proposto.

Na segunda parte será apresentada a documentação da implementação, descrevendo as interacções do protótipo, e de uma forma geral a sua estrutura de classes. De seguida apresentam-se os resultados obtidos e a sua análise.

6.1. Breve descrição da implementação

O protótipo foi implementado na linguagem JAVA, e de forma geral seguiu as funções definidas para o modelo. Foi seleccionada esta linguagem devido à sua filosofia de objectos/classes, portabilidade, e para permitir o reaproveitamento das estruturas existentes na linguagem, em termos de invocações remotas de classes – JAVA RMI – e implementações de cifras e métodos de autenticação.

Dado que o objectivo desta implementação é a realização de um *proof-of-concept* do modelo apresentado, restringiu-se o âmbito e o detalhe da mesma, tendo alguns pormenores sido relegados para uma implementação mais detalhada do modelo.

As componentes implementadas foram as seguintes:

- Foi implementado um sistema de controlo de acessos. No entanto, para efeitos do protótipo, este módulo apresenta um comportamento *dummy*, concedendo sempre acesso.
- Dois sistemas de autenticação: 1 tendo um comportamento *dummy*, sem qualquer tipo de autenticação, o outro utilizando autenticação NTLM, sendo opcional qual o sistema a ser utilizado.
- Um sistema de distribuição de chaves centralizado. A política adoptada para o protótipo foi gerar uma nova chave, sempre que há alteração da *membership* do grupo.
- Uma API cliente, que permite aos clientes interagirem com o servidor.
- Um servidor centralizado que fornece a abstracção necessária para corporizar a camada de segurança – criação de grupos seguros – e a interacção com o GCS, tendo sido utilizado o *spread* neste protótipo.

A componente de configuração do servidor – API do Servidor – foi prevista, mas não foi implementada, dado que foi considerada pouco relevante para o *proof-of-concept*. No entanto, o próprio código de

inicialização do servidor pode ser alterado para adicionar/remover parâmetros de configuração, como por exemplo: adicionar novos GKDCs, AS, ACS, e respectivas politicas.

6.2. Documentação da implementação do protótipo

A imagem apresentada a seguir representa as interacções possíveis entre o cliente, o servidor central e os seus componentes no protótipo sob a forma de diagrama, realizado com base nos diagramas de sequência do UML, mas com algumas alterações de nomenclatura.

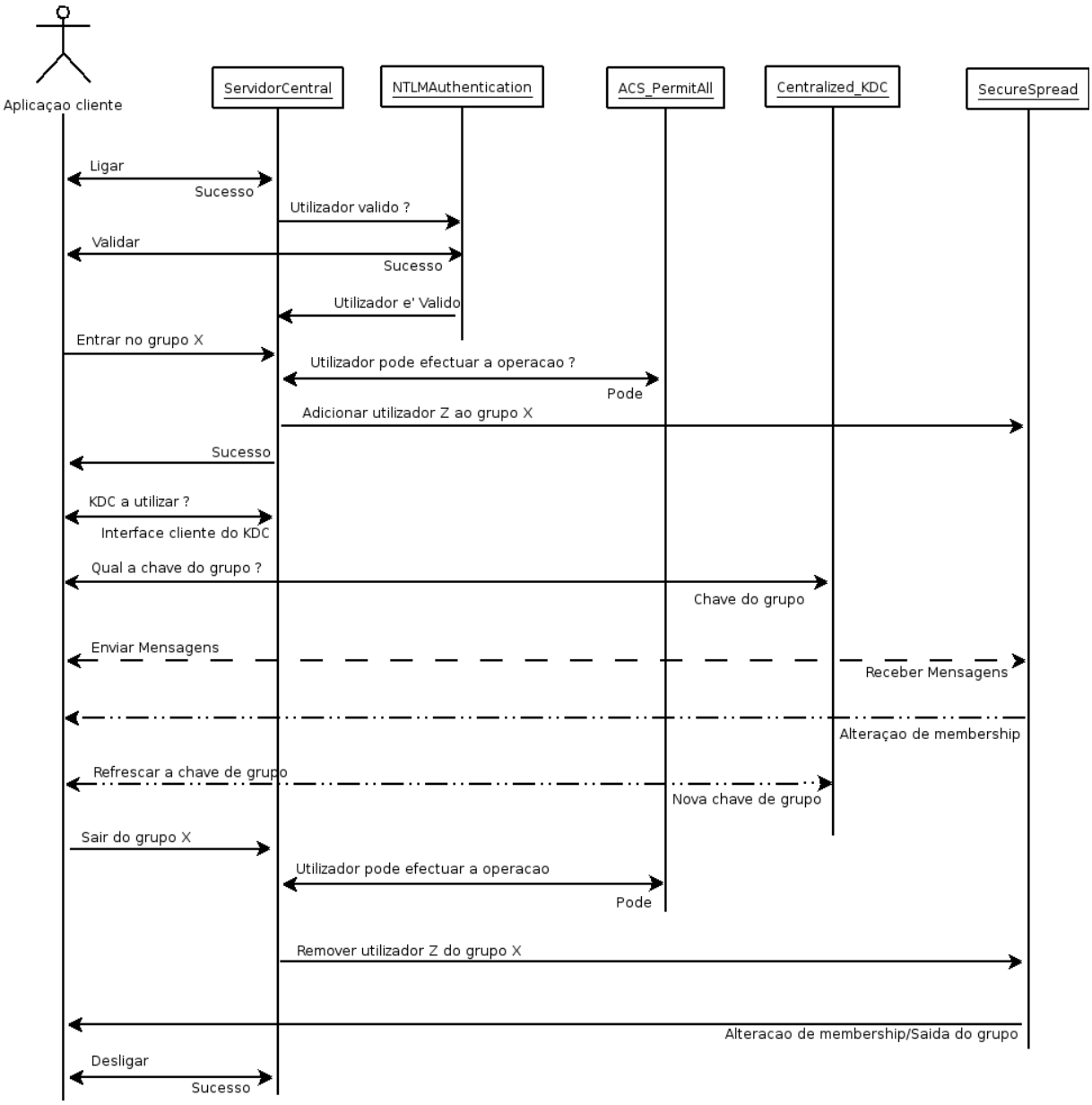

Figura 6-1 Diagrama de iterações do protótipo

Para permitir a realização do mapeamento entre o esquema lógico e os componentes do protótipo utilizados neste diagrama, estes são apresentados usando as designações com que aparecem na figura "5-3 – Esquema do modelo", onde é apresentada a representação lógica do modelo com base no qual foi implementado este protótipo.

Assim, o "actor" representa a aplicação cliente do protótipo, e cada uma das "interfaces" representam os seguintes componentes:

1.Cliente

2.Servidor Central

3.NTMLAuthentication – servidor de autenticação

4.ACS_PermitAll – servidor de controlo de acessos.

5.CentralizedKDC – servidor centralizado de geração e distribuição de chaves.

6.SecureSpread - servidor de comunicação em grupo.

Este diagrama tem um seguimento temporal e é composta por seis iterações distintas:

- Ligação ao servidor
- Entrar num grupo
- Envio e recepção de mensagens
- Alteração da membership
- Sair do grupo
- Sair do servidor

A operação de ligação ao servidor envolve a interacção com o servidor central e com o servidor de autenticação.

Um cliente apenas pode efectuar operações que envolvam grupos após efectuar a sua autenticação.

Para um cliente entrar num grupo (após ter efectuado a sua autenticação), este necessita de comunicar a sua intenção ao servidor central, obter a resposta do servidor e pedir a interface de comunicação com o servidor central de geração e distribuição de chaves. O ciclo de entrada no grupo termina com a instanciação da interface referida anteriormente e obtenção da chave de cifra a utilizar para a comunicação dentro do grupo.

Após ter sido efectuada a entrada no grupo, pode existir o envio e recepção de mensagens, bem como a recepção de uma mensagem especial – alteração de membership. Esta mensagem obriga ao refrescamento da chave do grupo.

Como estes são eventos opcionais (podem ou não acontecer durante o tempo que o cliente permanece no grupo), são utilizadas setas não continuas para os representar.

A operação de saída do grupo é comunicada ao servidor central e apenas é concluída com sucesso quando o cliente recebe a confirmação do servidor de comunicação.

A ultima iteração corresponde à sua saída do servidor.

Da implementação resultaram cinco classes utilizadas pelo servidor: uma classe responsável pelos dados da chave de um grupo (Centralized_KDC_Data), uma classe responsável pelo tratamento de mensagens do grupo (gcslisternerthread), uma classe que estabelece a ligação entre o cliente e o servidor (clientlistenerthread), uma classe que trata os pedidos do *re-keying* (clientprocessthread) e finalmente a classe servidor (Centralized_KDC).

O cliente é constituído por três classes: uma classe que contém a implementação do sistema de cifra (Centralized_KDC_Encryption_Implementation), a classe Centralized_KDC_Signing_Implementation que contém uma implementação do sistema de assinaturas digitais e a classe Centralized_KDC_Client que contém as classes anteriores.

É de referir por fim a classe (Centralized_KDC_Policy), que tem como objectivo descrever quais as políticas para os tamanhos da chave de cifra e de assinatura, bem como o tipo de chave de cifra e de assinatura digital utilizado. Por defeito é utilizado Blowfish com 56 bits e para assinatura é utilizado um HMAC.

No anexo I encontra-se, em detalhe, os diagramas das classes implementadas para a execução do prótotipo. Passamos agora à apresentação e análise dos resultados obtidos.

6.3. Resultados Obtidos

Neste capítulo apresenta-se alguns resultados obtidos durante uma utilização típica do protótipo e efectua-se a análise dos mesmos.

O ambiente de testes foi constituído por cinco máquinas, uma delas dedicada para servidor de comunicação (GCS) e para o servidor do protótipo. As restantes máquinas foram utilizadas para executar as aplicações cliente. A máquina servidor consiste num Intel Pentium IV a 2.4Ghz com 512MB RAM e os clientes são máquinas equipadas com Intel Pentium III, desde 600Mhz a 1Ghz, com os valores de memória RAM no intervalo 384-512MB.

De forma a tornar homogéneo o tempo de latência de rede, largura de banda e quantidade de tráfego, todas as maquinas foram ligadas a um *switch* dedicado.

A execução das aplicações cliente foi dividida proporcionalmente pelas 4 máquinas, com o objectivo de obter 16 conexões clientes, tendo cada máquina executado 4 aplicações. As execuções dos clientes foram iniciadas de forma rotativa e por ordem decrescente, da maquina mais rápida para a mais lenta (e com menos memória). Por exemplo, para a entrada de membros, a maquina mais rápida executou os clientes

1,5,9,13. Para a saída de membros a mesma maquina ficou com os clientes que saíram em 1º,4º,8º e 12º lugar, que correspondem ao 16º, 12º, 8º e 4º membros do grupo.

A aplicação cliente de teste foi utilizada para testar quatro situações:

1. Avaliação do tempo de entrada – *join* – de membros no mesmo grupo, e do tempo gasto na disseminação da chave de grupo para os membros.
2. Avaliação do tempo de saída de membros do grupo – *leave.*
3. Avaliação da primeira situação simulando um *mass join* – entrada em simultâneo de 4 membros.
4. Avaliação do tempo de um *mass leave* – saída em simultânea de 4 membros.

Estas situações foram identificadas como as que melhor evidenciam o comportamento do protótipo e que melhor permitem efectuar uma comparação com outros sistemas.

Por motivos temporais e de localização das várias máquinas, os tempos foram obtidos através de uma única execução das aplicações para as várias situações de análise. Após a obtenção dos tempos foi possível avaliar o desempenho do servidor em termos de escalabilidade e da problemática da inicialização de comunicações com os componentes do protótipo. Os tempos relativos à segunda situação permitem também analisar o impacto da geração e distribuição de chaves para o grupo.

As situações de *join* consistiram na entrada atómica/simultânea de um/quatro membros até atingir o número de membros comum a todas as situações - dezasseis.

As situações de *leave* tiveram inicio com dezasseis membros no grupo, tendo sido retirados conjuntos de um/quatro membros, até o grupo ficar vazio.

A medição dos tempos foi efectuada através da diferença de tempos do sistema. No caso da operação de *join*, estes foram medidos imediatamente antes da chamada do método no grupo e imediatamente após a obtenção da interface com a chave de cifra do grupo, sempre que um novo membro se juntou ao grupo. Apresenta-se de seguida um exemplo do código utilizado para medição:

```
TIME-START
    Join(<nome do grupo>);
    Set_GKDC_Optional_Data(<nome do grupo>, <nome do utilizador>);
    GKDC_Client_Interface gkdc_=.get_GKDC(<nome do grupo>);
TIME-END
```

O tempo de disseminação da chave foi obtido pela média ponderada da duração da chamada do método de refrescamento da chave pelos membros existentes no grupo, como podemos ver no exemplo que se segue:

TIME-START

GKDC_Encrypt_Implementation(<nome do grupo>).membership_change(<inteiro>);

TIME-END

O processo de interacção entre o cliente e o protótipo começa com o primeiro a estabelecer uma comunicação inicial com o servidor para transmissão de comandos. Nesta fase é efectuado o *handshake* inicial e em seguida a autenticação do utilizador. Depois de autenticado, o cliente executa o pedido de ligação ao grupo sendo permitido ou não o acesso ao grupo, conforme a politica de controlo de acessos. Depois de autorizado o acesso ao grupo pelo servidor, o pedido é enviado para o GCS onde é alterada a *membership* do grupo e, após ter entrado no grupo, efectua uma ligação SSL com o GKDC, onde irá obter e trocar as chaves de cifra do grupo. Esta ultima ligação corresponde à chamada do método GKDC_Client_Interface gkdc_=.get_GKDC(<nome do grupo>), referido no pseudo-código apresentado anteriormente.

Com os tempos obtidos da interacção de entrada no grupo, foi possível construir o seguinte gráfico:

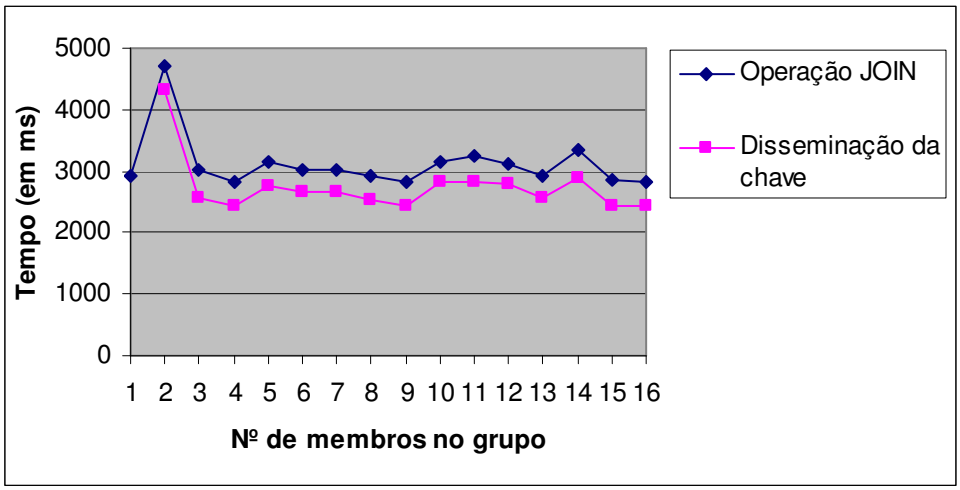

Figura 6-2 Gráfico de entrada atómica de membros

A linha azul corresponde ao tempo medido pelo cliente que entra no grupo, enquanto a linha rosa corresponde à média ponderada dos tempos observados pelo membros existentes no grupo para a disseminação da chave.

A disseminação da chave do grupo (*key refresh*) ocorre quando a sua *membership* é alterada e é avaliada pelo tempo que demora entre a alteração da *membership*, o estabelecimento da ligação SSL pelo novo membro ao servidor centralizado de geração e distribuição de chaves e a efectiva disseminação da chave para os restantes membros do grupo.

A conclusão imediata e mais notória que se pode obter deste teste é o tempo elevado da entrada de membros e da respectiva disseminação da chave. Por outro lado, à excepção do tempo obtido devido à entrada do segundo membro, apenas se notam pequenas variações ao tempo médio da entrada de membros.

O elevado tempo obtido resulta da soma do tempo de entrada no grupo com o tempo de refrescamento da *membership* do grupo, adicionado ao tempo de estabelecimento da sessão SSL, descrita anteriormente, e ao tempo de disseminação da nova chave.

É visível pelo gráfico que o tempo de *key refresh* é o factor que determina o tempo que demora um novo membro a entrar no grupo. Por sua vez esse valor é influenciado pelo tempo do estabelecimento da sessão SSL entre o novo membro e o *GKDC*.

Após o teste de entrada no grupo, procedeu-se ao teste de saída atómica de membros do grupo. Este teste tem como objectivo estudar o impacto da distribuição das chaves, tendo sido medido o tempo que cada aplicação cliente registou para a obtenção da nova chave de cifra do grupo.

Como já foi referido inicialmente, cada máquina executou 4 aplicações cliente, sendo cada cliente representado por uma linha no respectivo gráfico. A saída de membros foi efectuada da maquina mais rápida para a maquina mais lenta, de forma rotativa como descrito anteriormente.

Com os tempos obtidos foram construídos cinco gráficos, quatro referentes aos dados de cada máquina e um com as médias ponderadas do tempo de saída dos membros. Os gráficos de cada máquina contêm os tempos de refrescamento da chave observados por cada membro.

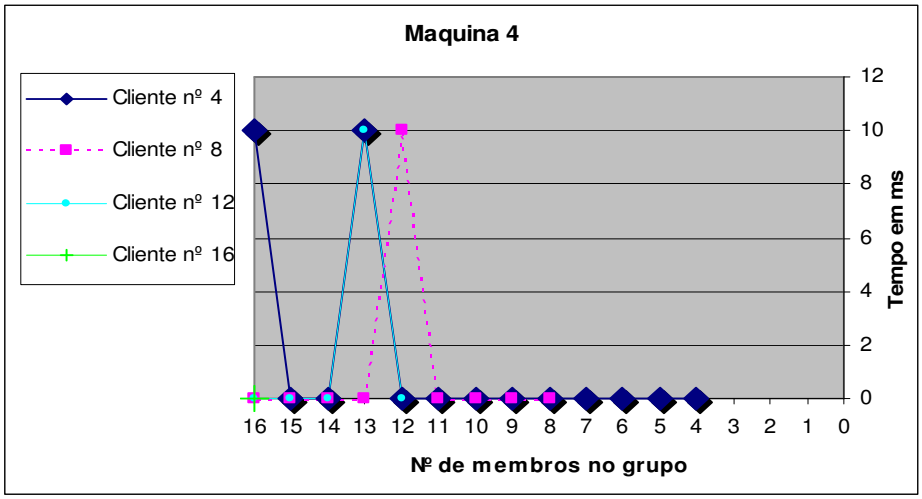

Figura 6-3 Gráfico de saída atómica de membros medidos pela máquina mais rápida

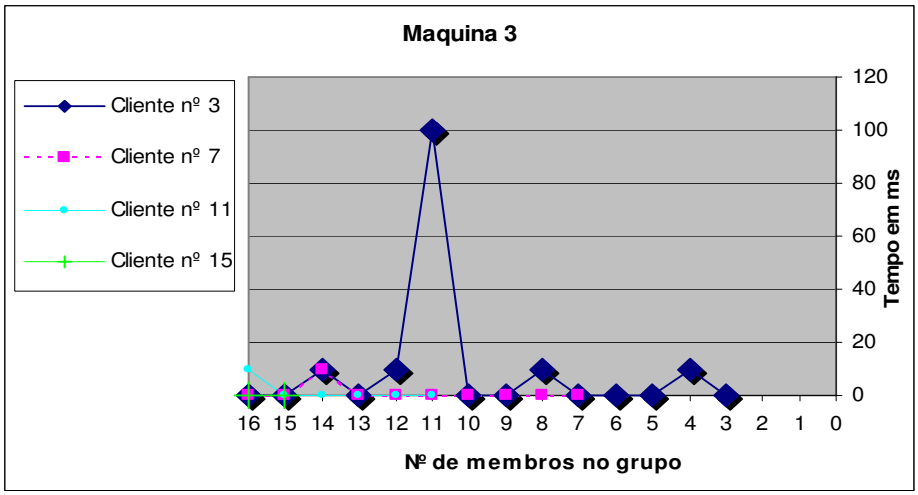

Figura 6-4 Gráfico de saída atómica de membros medidos por uma das máquinas com capacidade de processamento intermédio

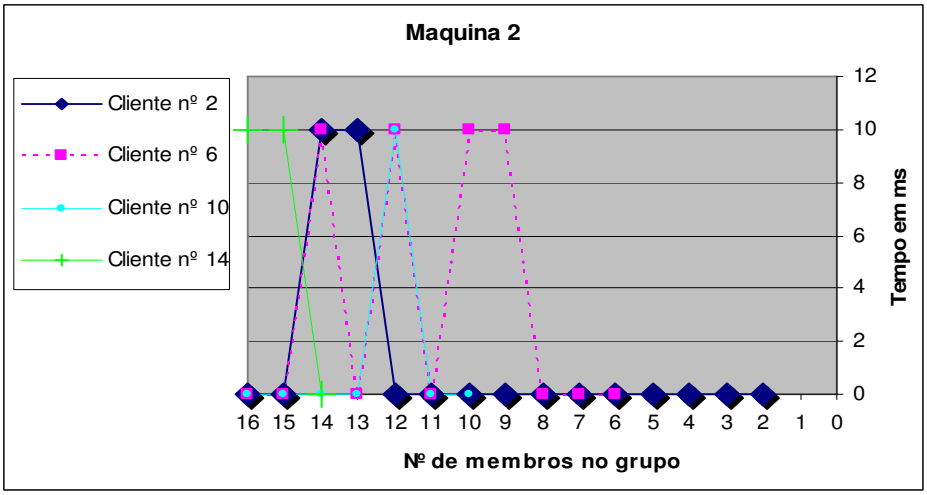

Figura 6-5 Gráfico de saída atómica de membros medidos por uma das máquinas com capacidade de processamento intermédio

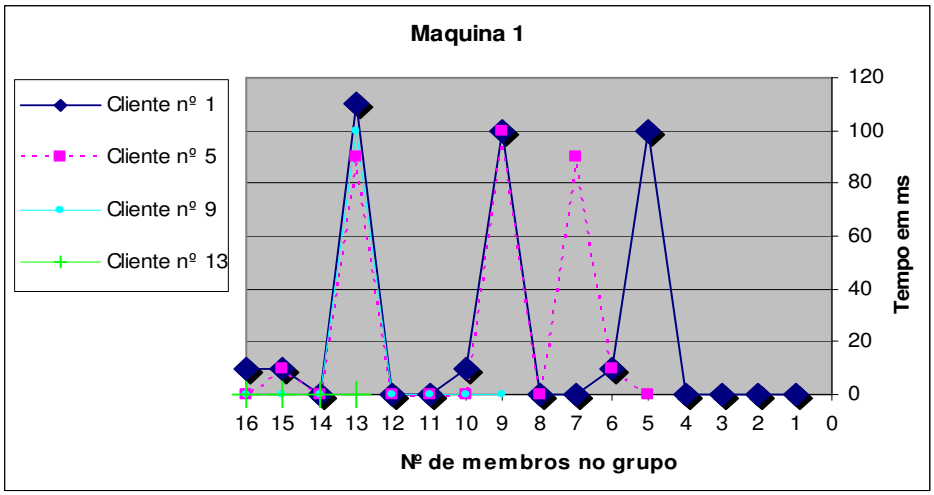

Figura 6-6 Gráfico de saída atómica de membros medidos pela máquina mais lenta

Destes quatro gráficos é possível identificar que, excepto para a maquina com capacidade de processamento mais baixa (ultimo gráfico), o tempo de distribuição de chaves é praticamente nulo. No caso desta maquina, o tempo de obtenção da nova chave é notoriamente mais elevado, o que revela a influência do processamento das máquinas clientes no desempenho global do protótipo.

Os tempos mais elevados foram obtidos em duas maquinas, maquina 1 e maquina 3, em situações idênticas. Quando um membro/cliente da respectiva maquina sai do grupo nota-se picos nos tempos. Para a maquina 3 isso é observado no cliente nº 11 e para a maquina 1 para os clientes nº 13, 9 e 5. Nesta ultima maquina observa-se também um pico para o cliente nº 7. Qual a relação do facto de este cliente residir na maquina 3 com o pico observado por ela própria para o seu cliente nº 11 é uma questão não analisada mas que deverá ser tida em conta em futuros resultados.

Um tempo máximo de 110 ms para a obtenção da nova chave confirma que o sistema permite escalabilidade sem degradar a *performamce* do grupo, já que em casos típicos de utilização na Internet este tempo é desprezável face ao tempo de comunicação entre os clientes e os servidores.

O próximo gráfico resulta da média ponderada dos tempos de saída dos membros. Os pontos mais elevados estão principalmente relacionados com a obtenção dos valores da maquina número um.

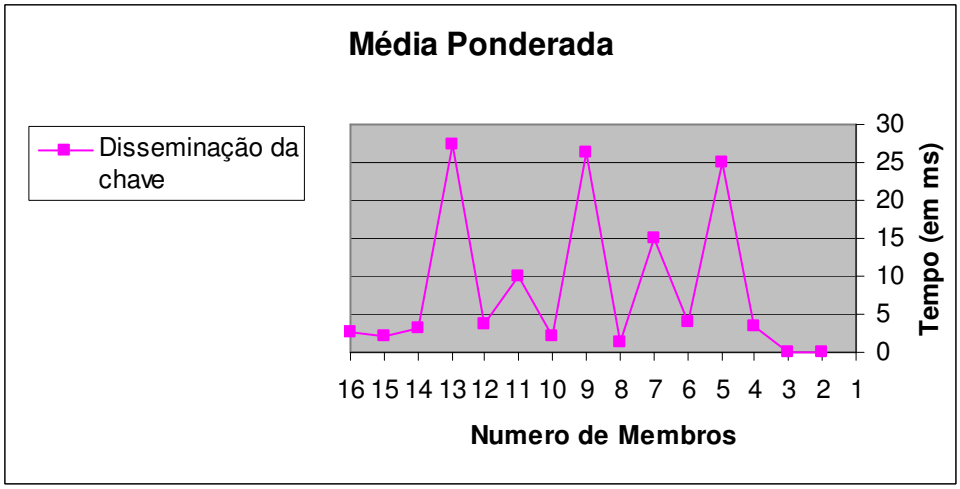

Figura 6-7 Gráfico da saída atómica de membros

Com uma média abaixo dos 30ms consideram-se os resultados bastante satisfatórios, revelando a potencial escalabilidade do protótipo.

Os últimos dois testes foram efectuados com os clientes a serem executados ao mesmo tempo em máquinas diferentes. O início da execução dos clientes ficou a cargo de uma tarefa agendada para o mesmo instante. Os relógios dos sistemas das maquinas encontravam-se sincronizados.

O objectivo da execução destes testes é simular um *mass join* e um *mass leave* por parte dos clientes. Os resultados obtidos também se poderão estender, respectivamente, para os caso de reunificação e partição de um grupo.

Os resultados obtidos para o *mass join* permitiu construir o seguinte gráfico:

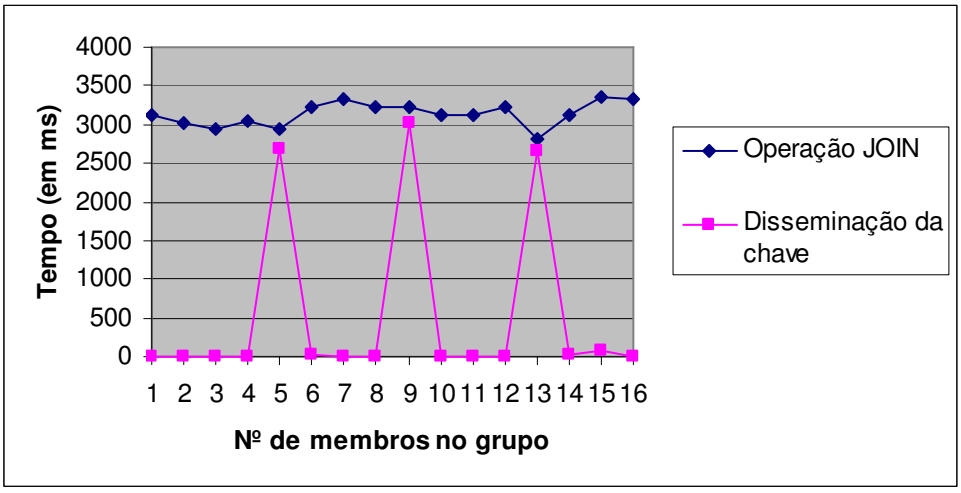

Figura 6-8 Gráfico de entrada simultânea de membros

Neste gráfico é imediatamente notório dois aspectos importantes. O primeiro aspecto é observar-se uma ligeira subida no tempo da operação JOIN ao grupo conforme aumenta o número de membros no grupo; o segundo aspecto são os três picos que se observa no tempo da disseminação da chave.

Relembra-se que a medição do tempo de disseminação da chave é efectuada pela média ponderada observada pelos membros existentes no grupo, isto é para o caso do membro número 5, a medição é efectuada pelos quatros membros já existentes no grupo.

A importância desta medição revela que quando temos um *mass join,* para os membros já existentes no grupo, o tempo de disseminação é totalmente influenciado pelo intervalo que os novos membros demoram para efectuar a negociação do canal seguro (SSL) com o servidor GKDC. Como há uma entrada simultânea de 4 elementos, para os membros já existentes no grupo o único tempo visível é o do primeiro elemento do novo conjunto a entrar; os tempos dos outros elementos são camuflados, pois todos os membros realizam a negociação do canal seguro com o servidor em paralelo. O resultado desta inicialização em paralelo dos clientes é visível no tempo de disseminação da chave de grupo, que é elevado para o quinto elemento e praticamente nulo para os restantes três elementos. Podemos concluir que em caso de reunificação de um grupo, o tempo máximo dessa reunificação está influenciada pela pior duração obtida quando todos os membros de um sub-grupo estabelecerem a comunicação segura com o GKDC.

Por outro lado, para os novos membros, o tempo de disseminação é praticamente nulo, pela mesma razão referida anteriormente, e o comportamento observado por estes membros é semelhante ao do teste de *leave.*

Para concluir os testes procedeu-se ao teste de *mass leave.* O gráfico que se apresenta de seguida reflecte a média ponderada obtida pelos membros do grupo, tal como foi descrito para o *leave* atómico. É interessante uma comparação entre este gráfico e do *leave* atómico, já que neste último gráfico não se detectam tempos ponderados maiores do que 5 ms – o maior tempo medido pelos clientes foi 10 ms.

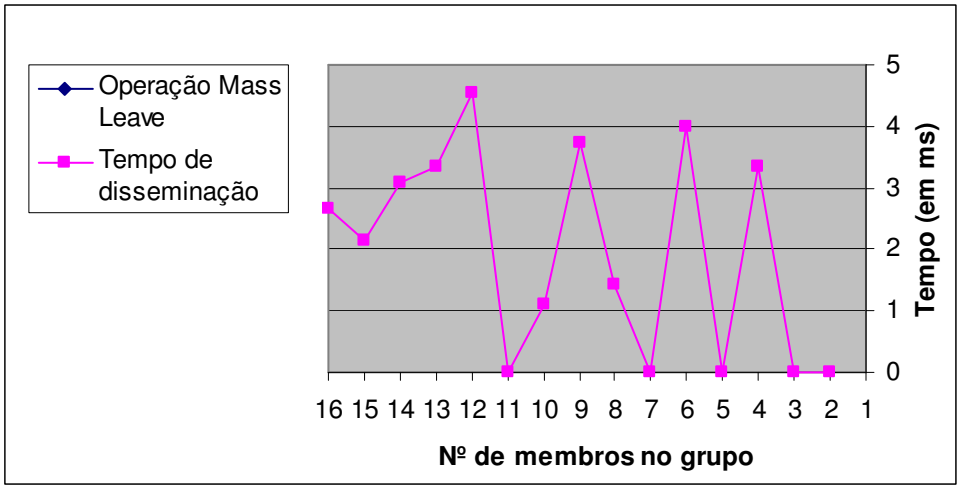

Figura 6-9 Gráfico de saída simultânea de membros

A conclusão da análise dos dois últimos gráficos revela que o protótipo se comporta melhor em situações de partição e reunificação de grupos. O estudo deste comportamento poderá ser uma matéria interessante para um trabalho futuro.

Para se obter uma maior percepção dos resultados obtidos, apresentam-se quatro gráficos obtidos por testes efectuados ao Secure Spread [66] utilizando um servidor centralizado de distribuição de chaves, tal como o que foi implementado para este protótipo, e o protocolo de gestão de chaves GDH referido no capítulo 4.

No teste efectuado ao Secure Spread foi utilizado um CKD em que o canal seguro entre o mesmo e o cliente foi baseado em Diffie-Hellman com chaves de 512 e 1024 bits. A análise do gráfico demonstra um incremento significativo do tempo da operação *join* quando se aumentou o tamanho da chave de 512 para 1024 bits. O comportamento do Secure Spread e do protótipo é semelhante, já que se nota um incremento no tempo das operações quando o número de clientes aumenta. Tal como no protótipo, e como era esperado, o tempo da operação *join* é totalmente indexado ao tempo do estabelecimento do canal seguro entre os clientes/membros e o servidor CDK.

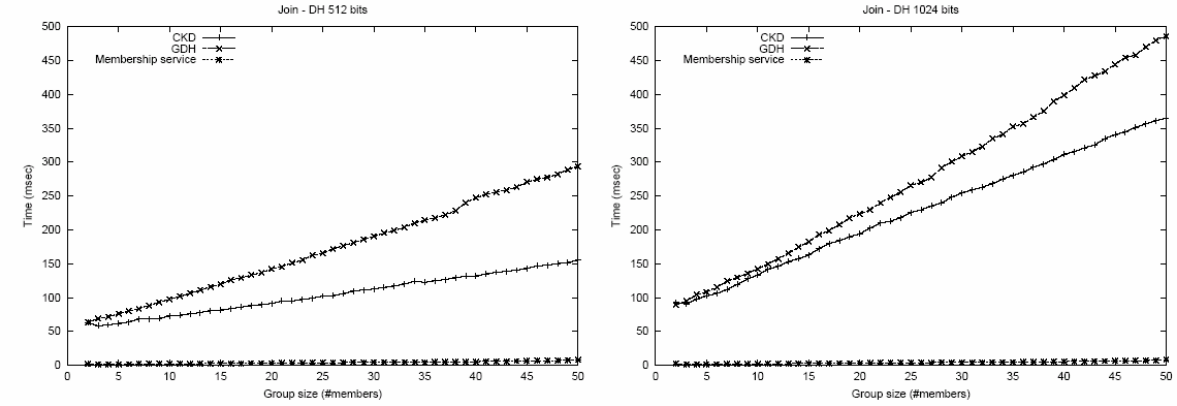

Figura 6-10 Gráfico de entrada atómica de membros no Secure Spread

Os valores obtidos para a saída de membros são semelhantes aos obtidos pelo protótipo, já que para o sistema o tempo médio mais elevado de saída de um membro foi de 110ms, um tempo bastante semelhante ao observados nos gráficos obtidos para o SecureSpread.

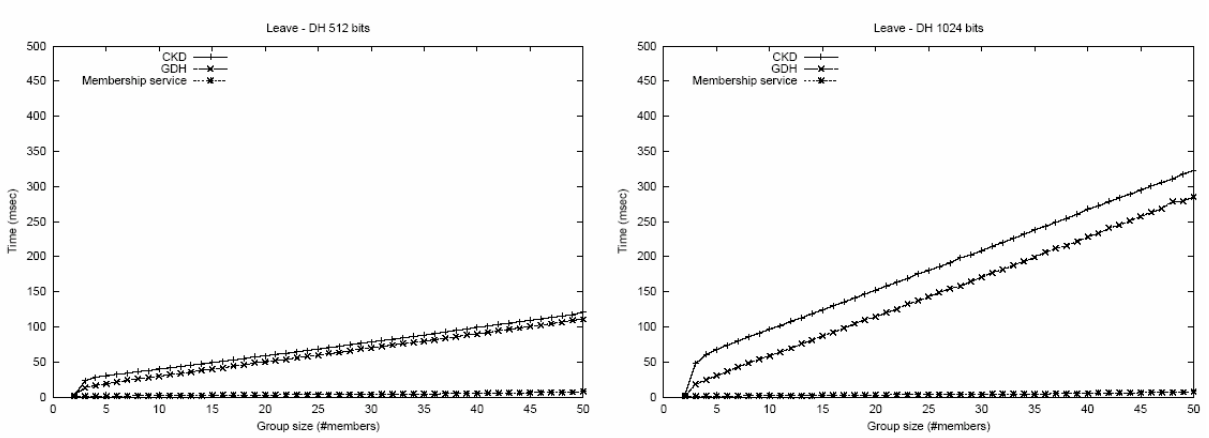

Figura 6-11 Gráfico de saída atómica de membros no Secure Spread

Desta forma comprova-se que a *performance* deste protótipo é equiparada à dos sistemas existentes.

Terá de ser levado em conta, no entanto, que poderão ser efectuadas melhorias. Apresentamos como exemplo o uso de protocolos de geração distribuída de chaves de grupo (como é o caso do GDH), para eliminar a necessidade do estabelecimento de um canal seguro com o GKDC.

Capitulo 7 – Conclusões

Com a evolução das necessidades de comunicação em grupo, foram sendo desenvolvidos vários SCG que suportam a criação de grupos dinâmicos de utilizadores e com tolerância a falhas. A evolução destes SCG produziram vários toolkits de programação, que apresentam no mínimo quatro operações base: juntar-se a um grupo, sair do grupo, enviar mensagens para um grupo e receber mensagens do grupo.

Os sistemas de ordenação das mensagens também sofreram evoluções significativas, passando da fase de nenhuma garantia na ordenação, à ordenação completa de todas as mensagens enviadas para o grupo.

No entanto, as evoluções realizadas tiveram sempre como objectivo as melhorias de desempenho e a adição de mais funcionalidades ao grupo.

O aspecto de segurança tem sido ignorado, tendo-se registado a evolução de outras questões como suporte e tolerância a falhas, e suporte a grandes latências, como pode ser o caso da Internet. No entanto, com o estabelecimento destas bases e com a evolução das necessidades de comunicação em grupo, recentemente o aspecto da segurança foi adoptado como objecto de estudo, estando ainda numa fase inicial.

A maioria dos sistemas criptográficos propostos e existentes têm-se provado seguros, mas têm como base um sistema de comunicação ponto a ponto. Muitos destes sistemas criptográficos não suportam grupos e a sua extensão para acomodar este requisito não é trivial, chegando mesmo a ser impraticável.

7.1. Trabalho Realizado

Nesta tese objectivou-se a problemática de adicionar e integrar serviços de segurança orientada para grupos em sistemas de comunicação fiável em grupo. O objectivo foi o estudo e a caracterização do suporte específico para autenticação em grupo, controlo de acessos em grupo e confidencialidade em grupo.

O estudo apresenta os métodos convencionais de segurança para ponto a ponto e quais as suas vantagens e desvantagens, bem como a sua evolução para comunicações multiponto.

Com base nessa análise e a partir do suporte genérico de um sistema de comunicação em grupo, propõe-se um modelo que integra um serviço de autenticação, um sistema de autorizações com mecanismo de controlo de acessos e um protocolo de gestão e distribuição de chaves criptográficas para sessões de comunicação em grupo.

Sempre com os princípios de segurança como base, este modelo permite converter qualquer sistema de SCG num sistema seguro de comunicação em grupo (SSCG), sendo essa a sua mais valia. O novo SSCG é customizavel conforme as necessidades específicas da utilização que se pretende efectuar, permitindo que

diferentes grupos tenham diferentes politicas de segurança para a autenticação, controlo de acessos e cifra das mensagens.

As vantagens deste modelo baseiam-se na sua fácil integração com sistemas de SCG já existentes, a possibilidade de adicionar novos serviços de controlo de acessos, autenticação, e sistemas de geração e distribuição de chaves. A modularidade do modelo tem benefícios quer para o programador quer para o utilizador final, já que uma interrupção em alguns dos serviços não obriga à interrupção do sistema como forma de comunicação segura em grupo.

Com o objectivo de demonstração da validade do modelo e para analise do mesmo, desenvolveu-se um protótipo.

A analise de desempenho do protótipo demonstra a validade do modelo e compara-o com um SSGC, o SecureSpread. Nesta analise fica patente o potencial que o modelo apresenta e quais poderão ser os próximos passos para a evolução do modelo.

7.2. Aspectos em aberto

Devido a questões temporais e logísticas, o modelo não foi testado até à sua exaustão. Através da sua implementação, este deve ser testado com um maior número de membros por grupos, bem como um maior número de grupos de forma a testar os limites do servidor. Com um maior número de testes, é possível optimizar a implementação e compara-la com mais modelos actuais de SSGC.

Também é importante perceber o impacto que os diferentes SCG podem ter no modelo e sua implementação.

A extensão do modelo de forma a suportar alta disponibilidade do servidor central, bem como dos seus componentes, é outro assunto que ficou em aberto e que poderá ser explorado como proposta de futuros trabalhos ou tema de dissertação.

7.3. Trabalho Futuro

A aplicação dos princípios de segurança a um sistema de comunicação em grupo levanta novos desafios, novos modelos e novos protocolos. Actualmente os conceitos que estão em estudo e que podem revolucionar a comunicação em grupo são os sistemas de assinaturas em grupo e a geração de chaves de forma distribuída e contributiva de forma eficiente.

A criação de politicas de segurança orientadas para comunicação em grupo deverão surgir com a utilização cada vez mais intensa deste tipo de comunicação. Estas politicas poderão implicar um novo componente para sua criação e distribuição a todos os servidores envolvidos.

A implantação do IPv6 como protocolo de comunicação permite melhorias para sistemas de comunicação em grupo. Essas melhorias anunciadas pelo protocolo devem ser estudadas e analisadas para se perceber como podem reforçar o modelo apresentado.

Estes (futuros) desenvolvimentos, devem-se reflectir no modelo apresentado, alterando de forma mínima a sua lógica e estrutura mas podendo alterar de forma significativa a comunicação entre os seus componentes.

Em suma, o modelo apresentado é susceptível de desenvolvimentos, como seja o uso de um sistema de controlo, disseminação das várias politicas para os módulos respectivos, sistemas de auditoria das políticas e utilização por partes dos utilizadores, e integração de sistemas de autenticação das mensagens originadas pelos componentes existentes.

Anexo I – Esquema de classes do protótipo

Foi criado um *package* principal, denominado mestrado, para a implementação do protótipo. Dentro deste foram incorporados 3 *packages*, cada um dizendo respeito ao seu componente.

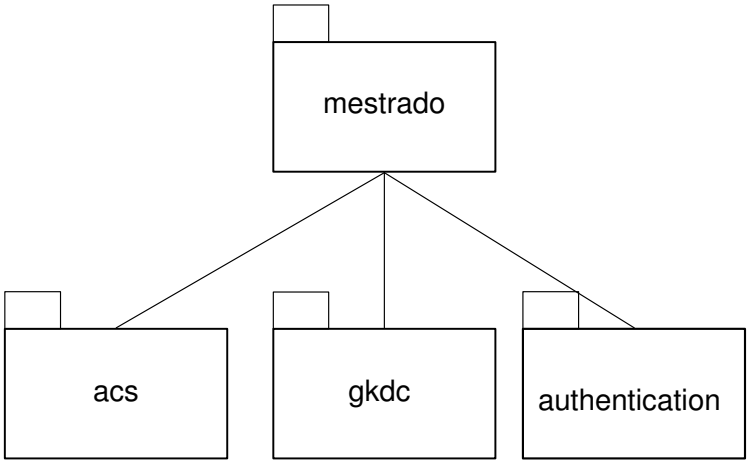

Figura i Estrutura de packages do protótipo

De seguida apresenta-se os diagramas de classes obtidos, com as interacções entre as várias classes que compõem o protótipo.

O diagrama de classes para o *package* mestrado (ignorando os outros *packages* nele incorporados) é o seguinte:

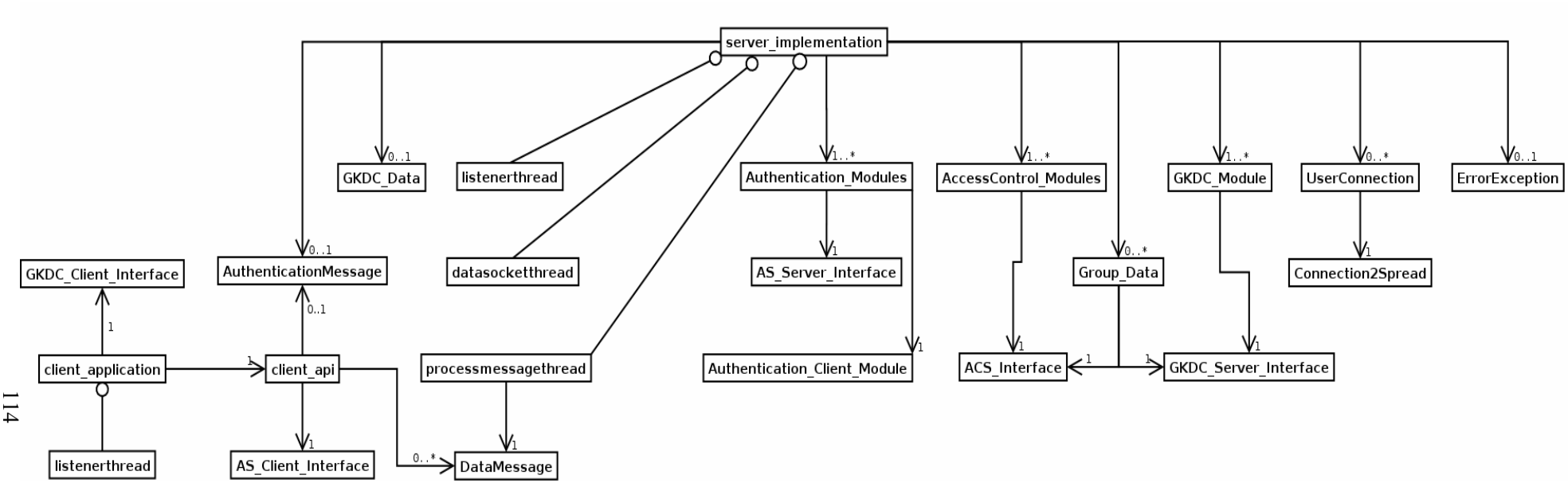

Figura ii Diagrama de classes do package mestrado

Neste *package* encontra-se a implementação do servidor e a API do cliente como classes principais. Passa-se agora a descrever as classes nele contidas:

- Server_Implementation – Classe principal do protótipo, oferece o serviço de segurança numa perspectiva de grupos.
- AuthenticationMessage – Usada para trocar mensagens de autenticação entre a API do cliente e os módulos responsáveis pela autenticação – Authentication_Modules.
- GKDC_Data – Usado para transmitir quais os GKDCs existentes no sistema.
- Authentication_Modules – classe que contém os modos de autenticação disponíveis para utilização. Utiliza as classes AS_Server_Interface para comunicar com o servidor de autenticação (*package authentication*), e a Authentication_Client_Module, que contém os dados a serem transmitidos para o cliente com informação sobre o método. A lista de métodos é assim enviada sob a forma de um vector de Authentication_Client_Module.
- ListenerThread, DataSocketThread e ProcessMessageThread – O ListenerThread e o DataThread destinam-se a servir os clientes. O *listener* é o responsável por atender os pedidos de operações dos clientes, o *DataSocket* escuta as mensagens de dados. Para cada mensagem recebida no *listener* é criado um ProcessMessage para a tratar, dado que existem mensagens que implicam a realização de operações bloqueantes.
- AccessControl_Module – utiliza a classe ACS_Interface e o seu objectivo é conter todos os modelos de controlo de acessos que possam ser utilizados pelo sistema.
- Group_Data – utiliza a classe ACS_Interface e GKDC_Server_Interface com o objectivo de especificar quais as respectivas implementações que o grupo utiliza.
- GKDC_Module – tal como a anterior, esta classe usa a classe GKDC_Server_Interface e contém todas as implementações de um GKDC que estão registadas no servidor.
- User_connection – utiliza a classe Connection2Spread e contêm os dados da ligação do principal ao servidor. A classe Connection2Spread é responsável pela integração do servidor com o *Spread*. Tem todas as funções que se pode efectuar com um SCG, tal como *connect, disconnect, join, leave, send* e *receive*.
- ErrorException – classe que serve para gerar excepções.
- Client_Application – aplicação para testar todo o sistema. Utiliza as classes ListenerThread, GKDC_Client_Interface e a Client_API. A classe ListenerThread é utilizada para receber mensagens, a classe GKDC_Client_Interface permite obter a implementação de uma classe para cifra de mensagens e por fim, a classe Client_API que tem como objectivo comunicar com o servidor e servir de API para o desenvolvimento de aplicações.

O *package* acs consiste em apenas uma classe, a classe ACS_PermitAll. Esta classe é uma implementação *dummy* de um sistema de controlo de acessos. Tal como o nome indica, esta implementação permite a execução de toda e qualquer operação de grupo.

O *package authentication* é constituído por duas implementações distintas. Uma implementação é *dummy*, onde não é efectuada qualquer tipo de autenticação, já que a classe autoriza sempre a entrada de um principal no sistema. A outra implementação efectua uma validação num sistema Microsoft Windows utilizando NTLM [58].

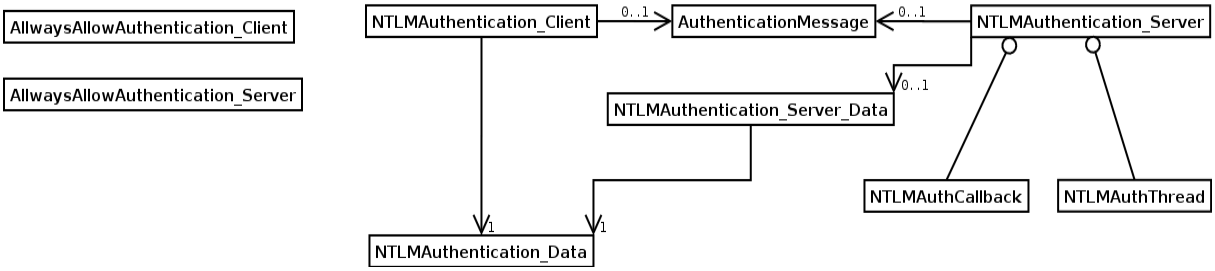

Figura iii Classes do sub package authentication

Na classe *dummy* AlwaysAllowAuthentication_X temos a componente de cliente e de servidor. A classe cliente limita-se a retornar de todas as operações. A classe servidor retorna sempre sucesso a todos os pedidos de autenticação.

O outro método de autenticação utiliza o serviço JAAS [59] do Java e é baseado em NTLM. Está dividida na classe cliente – NTLMAuthentication_Client e nas classes servidoras que têm como objectivo efectuar a autenticação. A autenticação é efectuada na máquina onde se encontra em execução o servidor de autenticação. Para tal utiliza a classe AuthenticationMessage, que permite efectuar a troca das mensagens de autenticação entre o cliente e o servidor. Utiliza também um *thread* para receber os dados dos principais e uma outra classe, denominada NTLMAuthentication_Server_Data, que permite guardar os dados enviados pelos clientes e efectuar a autenticação quando for solicitado.

O último sub *package*, gkdc, é constituído pela implementação de um servidor centralizado de distribuição de chaves.

Esta implementação está dividida nas secções cliente e servidor. O servidor é responsável pela geração da chave simétrica que o grupo utiliza. Na altura de criação do grupo e caso seja este o gkdc seleccionado pelo principal, o servidor junta-se ao grupo e, sempre que houver uma alteração à *membership* do grupo, é criada uma nova chave para o grupo. É da responsabilidade do cliente, através de funções fornecidas nas implementações da classe de cifra e assinatura, obter a nova chave do grupo.

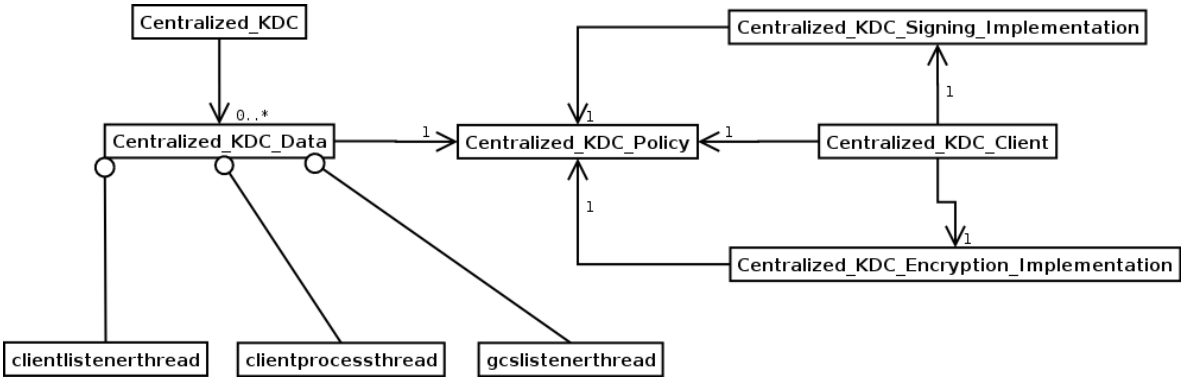

Figura iv Diagrama de dependências da implementação de um GKDC

Bibliografia

[1] Moser, L. E., Amir, Y., Melliar-Smith, P. M., and Agarwal, D. A. Extended Virtual Synchrony. In *Proceedings of the IEEE 14th Internacional Conference on Distributed Computing Systems* (Poznan, Poland, June). IEEE Computer Society Press, Los Alamitos, CA, 56-65.

[2] K. P. Birman and T.A. Joseph, "Exploiting virtual synchrony in distributed systems," *Proceedings of the ACM Symposium on Operating System Principles* (1987), pp. 123-138.

[3] K.P. Birman, "Virtual synchrony model," In: Reliable Distributed Computing with the Isis Toolkit, IEEE Press.

[4] L. Lamport, "Time, clocks, and the ordering of the events in a distributed system," *Communications of the ACM* (July 1978), pp. 558-565.

[5] B. Preneel, *Analysis and Design of Cryptographic Hash Functions*, Ph.D. Thesis, Katholieke University Leuven, 1993

[6] M.J.B. Robshaw, *MD2, MD4, MD5, SHA and Other Hash Functions*, Technical Report TR-101, version 4.0, RSA Laboratories, 1995

[7] National Institute of Standards and Technology. *Data Encryption Standard, Federal Information Processing Standards Publication (FIPS PUB) 46-1*. U.S. Department of Commerce, 1977. Reaffirmed 1988.

[8] American National Standards Institute. *Data Encryption Algorithm, ANSI X3.92-1981*, 1980.

[9] J. L. Massey and X. Lai. Device for converting a digital block and the use thereof. International Patent PCT/CH91/00117, November 1991.

[10] B. Schneier, *Fast Software Encryption, Cambridge Security Workshop Proceedings (December 1993)*, Springer-Verlag, 1994, pp. 191-204.

[11] National Institute of Standards and Technology. *Data Encryption Standard, Federal Information Processing Standards Publication (FIPS PUB) 46-3*. U.S. Department of Commerce. Reaffirmed 1999.

[12] National Institute of Standards and Technology. *Advanced Encryption Standard (AES), Federal Information Processing Standards Publication (FIPS PUB) 197*. U.S. Department of Commerce, 2001.

[13] R.L.Rivest, A.Shamir, L.M.Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM, Feb 1978

[14] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12), December 1978.

[15] D. Denning and G. Sacco. Timestamps in key distributed protocols. *Communication of the ACM*, 24(8):533--535, 1981.

[16] W. Diffie and M. Helman. New directions in cryptography. *IEEE Transactions on Information Society*, 22(6):644--654, november 1976.

[17] J. G. Steiner, B. Clifford Neuman, and J.I. Schiller. Kerberos: An Authentication Service for Open Network Systems. In *Proceedings of the Winter 1988 Usenix Conference*. February, 1988. (Version 4).

[18] Alan O. Freier, Philip Karlton and Paul C. Kocher. The SSL Protocol version 3.0, Internet Draft, Transport Layer Security Working Group, November 1996.

[19] CCITT, *Recommendation X.509: The Directory Authentication Framework*, 1988.

[20] P. Zimmermann, Pretty good privacy. http://www.pgpi.org/

[21] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of applied cryptography*, CRC Press series on discrete mathematics and its applications. CRC Press, 1997, ISBN 0-8493-8523-7.

[22] G. Ateniese, M. Steiner and G. Tsudik, *New Multi-party Authentication Services and Key Agreement Protocols* on IEEE Journal on Selected Areas in Communication, May 2000.

[23] Y. Kim, A. Perrig and G. Tsudik, *Simple and Fault-Tolerant Key Agreement for Dynamic Collaborative Groups*, ACM CCS 200, November 2000.

[24] Y. Kim, A. Perrig and G. Tsudik, *Communication-Efficient Group Key Agreement*, IFIP SEC 2001, Jun. 2001.

[25] *Multicast* Security Group. http://www.ietf.org/html.charters/msec-charter.html

[26] L. Rivest, Ado Shamir and Yael Tauman, *How to Leak A Secret*. In ASIACRYPT, 2001

[27] Y. Amir and J. Stanton, *The Spread Wide Area Group Communication System*, Technical Report CNDS-98-4, The Center for Networking and Distributed Systems, The Johns Hopkins University.

[28] Flush Spread. http://www.cnds.jhu.edu/research/group/flush_spread/

[29] Y. Amir, C. Nita-Rotaru and J. Stanton, *Framework for Authentication and Access Control of Client-Server Group Communication Systems* In the Proceedings of the Third Internation Workshop on Networked Group Communication, London, UK November 7-9 2001.

[30] M. Nystrom, *The SecurID SASL mechanism*, RFC-2808, April 2000.

[31] V. Samar and R. Schemers, *Unified login with Pluggable Authentication Modules (PAM)*, OSF-RFC 86.0, October 1995.

[32] Y. Amir, G. Ateniese, D. Hasse, Y. Kim, C. Nita-Rotaru, T. Schlossnagle, J. Schultz, J. Stanton and G. Tsudik, *Secure Group Communication in Asynchronous Networks with Failures: Integration and Experiments*, IEEE ICDCS 2000, April 2000.

[33] U. Government, *Data encryption standard*, Technical Report 46, National Bureau of Standards, Federal, 1977.

[36] R. Rivest, *The md5 message digest algorithm,* RFC 1321, SRI Network Information Center, 1992.

[37] Reliable *Multicast* Protocol. http://www.ietf.org/html.charters/rmt-charter.html

[38] S. Armstrong, A. Freier, K. Marzullo, RFC 1301, *Multicast* Transport Protocol, 1992.

[39] O. Pereira, J.-J. Quisquater, *Some Attacks upon Authenticated Group Key Agreements Protocols,* To appear in Journal of Computer Security, 2002.

[40] O. Pereira, *Modelling and Security Analysis of Authenticated Group Key Protocols.* 2003

[41] R.V. Renesse, K. P. Birman and S. Maffeis, *Horus, a flexible group communication system,* Communications of the ACM, April 1996.

[42] O. Rodeh, K. P. Birman and D. Dolev. *Optimized groiup rekey for group communications systems.* In Symposium Network and Distributed System Security, Febuary 2000.

[43] O. Rodeh, K. P. Birman and D. Dolev. *A study of group rekeying.* Technical Report TR2000-1791, Cornell University Computer Science, March 2000.

[44] O. Rodeh, K. P. Birman and D. Dolev. *The Architecture and Performance of Security Protocols in the Ensemble Group Communication System*, Journal of ACM Transactions on Information Systems and Security (TISSEC), October 2000.

[45] P. McDaniel, A- Prakash, and P. Honeyman, *Antigone: A Flexible Framework for Secure Group Communication*, In Proceedings of the 8th USENIX Security Symposium, pages 99-114, August 1999.

[46] G. Ateniese, M. Steiner and G. Tsudik, New Multi-party Authentication Services and Key Agreement Protocols, In IEEE Journal on Selected Areas in Communication, May 2000.

[47] National Institute of Standards and Technology (NIST), *Announcement of Weakness in the Secure Hash Standard*, 1994

[48] K. Birman, T. Joseph, Reliable Distributed Computing with the Isis Toolkit, IEEE Computer Society Press, 1994

[49] L. Moser, P. Melliar-Smith, D. Agarwal, R. Budhia, A. Papadopoulus, Totem: A Falt-Tolerant *Multicast* Group Communication System, in Communications of the ACM, April 1996

[50] Y. Amir, D. Dolev, S. Kramer, D. Malki, Transis: A Communication Sub-System for High Availability, in FTCS Conference, July, 1992

[51] K. Berket, L. Moser, P. Melliar-Smith, The Intergroup Protocols: Scalable Group Communication for the Internet, in proceedings of the IEEE Globecom'98, 1998

[52] O. Babaoglu, R. Davoli, A. Montresor, Partitionable Group Membership: Specification and Alghoritms, TR UBL CS97-1, Dep. Of Computer Science, University of Bologna, January 1997

[53] M. Reiter, K. Birman, L. Gong, Integrating Security in a Group Oriented Distributed System, in proceedings of the 1998 International Conference on Parallel and Distributed Processing Techniques and Applications, pp 1591-1598, July, 1998

[54] C. Malloth, A. Schipper, View Synchronous Communication in Large Scale Networks, in proc. 2nd Open Workshop of the Esprit Broadcast Project, July, 1995

[55] R. Canneti, B. Pinkas, A Taxonomy of *Multicast* Security Issues, draft-irtf-smug-taxonomy-00.txt, April 1999

[56] S. Kent, R. Atkinson, Security Architecture for the Internet Protocol, RFC 2401, November 1998

[57] M. Hayden, *The Ensemble System* Cornell, University Technical Report, TR98-1662, January 1998

[58] NTML. http://davenport.sourceforge.net/ntlm.html

[59] JAAS. http://java.sun.com/products/jaas/

[60] Security Architecture for Open Systems Interconnection, ITU-T, http://www.itu.int/rec/recommendation.asp?type=items&lang=E&parent=T-REC-X.800-199103-I, (Mar, 1991), com aditamento X.800-A1 (Oct, 1996)

[61] FC 2828, R. Shirey, RFF 2828 - Internet Security Glossary, Internet Engimeering Task Force – Network Security Group, May, 2000.

[62] C. Pfleeger, "Security in Computing", Upper Saddle River, Prentice Hall, 1997.

[63], R. Nichols, "ICSA Guide to Cryptography", McGraw Hill, 1999

[64], Charlie Kaufman, R. Perlman, M. Speciner, "Network Security: Privae Communication in a Public World", 2nd Ed., Prentice Hall, ISBN 0-13-046019-2, 2002

[65], William Stallings, "Network Security Essentials: Applications and Standards", 2nd Edition, Prentice Hall. ISBN 0-13-035128, 2003

[66], Y. Amir, Y. Kim, C. Nina-Rotaru, J. Schultz, J. Stanton, G. Tsudik, "Secure Group Communication Using Robust Contributory Key Agreement", in proceedings of the IEEE Transactions on Parallel and Distributed Systems, 2004.

[67], D. Wallner, E. Harder, R. Agee, "Key Management for Multicast: Issues and Architectures", RFC 2627, June 1999.

[68], M. Waldvogel, G. Caronni, D. Sun, N. Weiler, B. Plattner, "The VersaKey framework: Versatile group key management", IEEE – Selected Areas in Communications, September 1999.

[69], D. McGrew, A. Sherman, "Key establishment in large dynamic groups using one-way function trees", TIS Labs at Network Associates Technical Report No. 0755, May 1998.

[70], R. Canetti, T. Malkin, K. Nissim, "Efficient communication-storage tradeoffs for mulitcast encryption", Advances in Cryptology – EUROCRYPT '99, 1999.

[71], A. Perrig, D. Song, J. Tygar, "ELK, A new protocol for efficient large-group key distribution", Proceedings of the IEEE Symposium on Security and Privacy, 2001.

[72], A. Ballardie, "Scalable Multicast Key Distribution", RFC 1949

[73], H. Harney, C. Muckenhirn, "Group Key Management Protocol (GKMP) Specification", RFC 2093.

[74], S. Mittra, "Iolus: A framework for scalable secure multicasting", Proceedings of the ACM SIGCOMM, 1997.

[75], L. Dondeti, S. Mukherjee, A. Samal, "Scalable secure one-to-many group communication using dual encryption", Computer Communications, 17 November 1999.

[76], B. Briscoe, "MARKS: Multicast key management using arbitrarily revealed key sequences", Procedings of the 1st International Workshop on Network Group Communication", November 1999.

[77], R. Molva, A. Pannetrat, "Scalable multicast security in dynamic groups", Proceedings of the 6th ACM on Computer and Communications Security, November 1999.

[78], B. DeCleene, L. Dondeti, S. Griffin, T. Hardjono, D. Kiwior, J. Kurose, D. Towley, S. Vasudevan, C. Zhang, "Secure group communications for wireless networks", Proceedings of the MILCOM, June 2001.

[79], S. Setia, S. Koussih, S. Jajodia, "Kronos: A scalable group re-keying approach for secure multicast", Proceedings of the IEEE Symposium on Security and Privacy, 2000.

[80], M. Burmester, Y. Desmedt, "A secure and efficient conference key distribution system (extend abstract)", Advances in Cryptology - EUROCRYPT 94, 1994.

[81], S. Rafaeli, D. Hutchison, "Hydra: A decentralized group key management", Proceedings of the 11th IEEE Interrnational WETICE: Enterprise Security Workshop, 2002.

[82], C. Becker, U. Willie, "Communication complexity of group key distribution", Proceedings of the 5th ACM Conference on Computer and Communications Security, 1998

[83], C. Boyd, "On key agreement and conference key agreement", Proceedings of the Information and Security and Privacy: Australasian Conference, 1997

[84], O. Rodeh, K. Birman, D. Dolev, "Optimized group rekey for group communication systems", In Network and Distributed System Security, 2000.

[85], L. Dondeti, S. Mukherjee, A. Samal, "A distributed group key management scheme for secure many-to-many communication, Techinal Report PINTL-TR-207-99 Department of Computer Science, University of Maryland, 1999

[86], Y. Kim, A. Perrig, G. Tsudik, "Simple fault-tolerant key agreement for dymanic collaborative groups", In Proceedings of the 7th ACM Conference in Computer and Communication Security, 2000.